Complex Systems **6** (1992) 179–191

Habituation in Learning Vector Quantization

Tamás Geszti
István Csabai*
Atomic Physics Department, Eötvös University,
Puskin u. 5-7, H-1088 Budapest, Hungary

Abstract. A modification of Kohonen's Learning Vector Quantization is proposed to handle hard cases of supervised learning with a rugged decision surface or asymmetries in the input data structure. Cell reference points (neurons) are forced to move close to the decision surface by successively omitting input data that do not find a neuron of the opposite class within a circle of shrinking radius. This simulates habituation to frequent but unimportant stimuli and admits problem solving with fewer neurons. Simple estimates for the optimal shrinking schedule and results of illustrative runs are presented.

1. Introduction

In practical applications of neural computation with supervised learning it seems to be a serious problem that dominant features of the training set, although learned almost immediately, continue occupying computational resources and thereby inhibit the learning of finer details.

Living organisms are capable of managing that difficulty by means of a kind of non-associative learning called *habituation* [1, 2]: the ability of the nervous system to learn how to ignore frequent and strong, but unimportant, stimuli. We suggest that in many situations it can prove helpful to look for analogous strategies in machine learning.

The aim is not new. Principal component analysis, a method of mathematical statistics easily implemented by neural algorithms [3–8], consists of a systematic decomposition of the training data set into orthogonal principal components in succession. After finding the strongest component, one projects into a subspace orthogonal to it to find the next strongest, and so forth.

In the present paper we want to add a shift of viewpoint to that approach, saying that under some circumstances the most essential feature of principal component analysis is not the way to find the strongest features—any poor

*Electronic mail address: eapd001@hueco.uni-wien.ac.at

method is good enough to do that—but the subsequent projection, allowing one to find less pronounced details in the orthogonal subspace. Viewed from this perspective, the procedure might be called Principal Component Habituation.

Orthogonal component decomposition is not the only possible strategy. Analogous in scope is the work of Kinzel and Ruján [9], who suggest restricting the training of a simple perceptron to patterns orthogonal to the actual decision (or weight) vector. This results in an improvement of the generalization ability of the perceptron (see also [10]). In light of the above reasoning, the actual decision vector in their approach served strictly as a statistical estimate to the strongest principal component.

Genetic algorithms provide another way to approach the problem (for a closely related discussion see [11]). What should undergo evolution (Darwinian or not) in the present case are selected subsets of the training set, of a growing fitness in teaching good classification. In this sense, unimportant subsets are less fit, and they are doomed to perish by habituation.

Let us briefly describe two examples from our own recent work, where *ad hoc* methods proved useful in handling analogous difficulties.

Separation of quark and gluon jets. Jets, detected in high-energy particle accelerators, are bunches of particles emerging from the same point within a narrow angular range. Those originating from a quark and those from a gluon (both invisible) are similar, but a statistical analysis can detect features that distinguish them. One of the features is dominant: it is total energy of the jet, furnishing a rough classification. We had to restrict a neural network analysis to subclasses of equal energy to improve performance [12].

Detecting signal peptides. Signal peptides are short terminal segments temporarily attached to protein sequences as a postal code to conduct them from the place of synthesis to the place of utilization. In trying to tell whether a given amino acid sequence contains a signal peptide, some dominant features concerning the chemistry of the joint between signal and signaled chain had been recognized, furnishing a rough classification. We had to use these features as pre- and post-filtering sets to improve performance of a neural classifier [13].

In the following, we use systematic selection from the training input set, with the clearly stated aim of implementing habituation. We have found it particularly convenient to start from Kohonen's Learning Vector Quantization (LVQ) method [14, 15], in which our approach has a transparent geometrical meaning. LVQ is summarized in its original form in section 2, presenting a physically motivated estimate of the computation time needed to complete learning. Section 3 introduces our modification of LVQ by adding simulated habituation; some typical situations related to rugged decision surfaces or asymmetric data structures and calling for our approach are also discussed there. In section 4 we give simple estimates about the expected performance and limitations of our method. It will be shown that the basic effect of our procedure is to reduce the number of neurons needed to solve a classification task. In some cases it speeds up learning; in all cases it speeds up doing the

learned classification. Section 5 describes the first numerical tests on simple model tasks and summarizes our experience.

2. Learning Vector Quantization

Vector quantization is a generalization of analog-to-digital conversion to vector inputs $\mathbf{x}$, "quantized" into the closest one of a set of predefined discrete values $\mathbf{w}_i$—closeness being defined according to euclidean, Manhattan, or any reasonable distance. That divides the input space into Voronoi cells, the ith of which contains all points $\mathbf{x}$ to be quantized into the same cell center $\mathbf{w}_i$. Cell centers themselves can be visualized as synaptic connection strengths multiplexing input $\mathbf{x}$ toward neurons labeled by $i = 1, \ldots, N$. The neurobiological background of this assignment implies a layer of neurons competing by lateral inhibition; on the arrival of an input, after some transient time only the "winner" will give an output signal. We will often refer to $\mathbf{w}_i$ as the position of neuron i in the input space, and think of neurons as quasi-atoms moving in an adaptation process (see below).

Neurons can be bunched together to represent larger classes (or categories) of inputs. Kohonen's LVQ turns this tool into an adaptable classifier by adding a learning rule. That modifies the connection strengths—visually it displaces neurons in the input space—so predefined classes of neurons provide an optimal representation for sets of different kinds of inputs. These sets are often noisy, therefore they can overlap.

Kohonen's LVQ rule [14, 15] consists of three steps:

1. choose (randomly or not) an element $\mathbf{x}$ of class q of the training set;
2. find the closest neuron i (the "winner")—the one for which the distance $|\mathbf{x} - \mathbf{w}_i|$ is the smallest—and determine class e to which neuron i belongs;
3. modify the connection strength $\mathbf{w}_i$ of the winner according to

$$\mathbf{w}_i \quad \rightarrow \quad \mathbf{w}_i + S(e,q)\lambda(\mathbf{x} - \mathbf{w}_i), \tag{1}$$

where λ is the amplitude of learning, and $S(e,q)$ is $+1$ if neuron class e represents input class q and -1 otherwise. In practical applications λ is usually diminished gently in time as adaptation proceeds (see below in section 5).

Equation (1) can be interpreted by saying that neurons are attracted to training patterns of the same class and repulsed by those of other classes.

If there are just two classes, one can label them $q = \pm 1$ for training set classes and $e = \pm 1$ for the corresponding neuron classes; then $S(e,q) = eq$. If, as mentioned above, neurons are regarded as quasi-atoms, e and q are analogous to electric charges of neurons and patterns, respectively (however, charges of equal sign attract each other).

Since one of the main points of the present paper is to show a way to speed up LVQ in some hard cases, we need a rough estimate of the computing time needed for the original LVQ. To this end we use the quasi-atom visualization (see also [16], where more details about LVQ dynamics are discussed).

The computing time is determined by the last and slowest stage of the learning process: a diffusion-like homogenization of N neurons of a given class over a range of volume V of the d-dimensional input space, dominated by inputs of the respective class. If the input range is neither too oblong nor too oblate (our suggestions presented below in section 3 are less useful in those cases), it can be approximated by a hypersphere and its largest diameter L can be estimated from $V = A_d(L/2)^d$. Here we do not need an explicit expression for the coefficient A_d. Then, for the last stage when inhomogeneities are already smooth, one can use a diffusion equation to evaluate the homogenization time $(L/2\pi)^2/D$, where D is the diffusion coefficient of neuron quasi-atoms. With the above estimate of L this gives

$$t_{\mathrm{LVQ}} = \pi^{-2} D^{-1} \left(V/A_d\right)^{2/d}. \tag{2}$$

We still need an estimate of D. Let us start by approximating the Voronoi cell around a given neuron by a hypersphere of radius R_{cell} to be determined from its volume $A_d R_{\mathrm{cell}}^d = 1/\varrho$, where ϱ is the local density of neurons. Diffusion is driven by its gradient. Then it is easy to see that the Voronoi hypersphere would swell toward the low-density side and contract from the high-density side; this is equivalent to shifting the whole cell by $\delta = R_{\mathrm{cell}} \nabla R_{\mathrm{cell}} = R_{\mathrm{cell}}(dR_{\mathrm{cell}}/d\varrho)\nabla\varrho$.

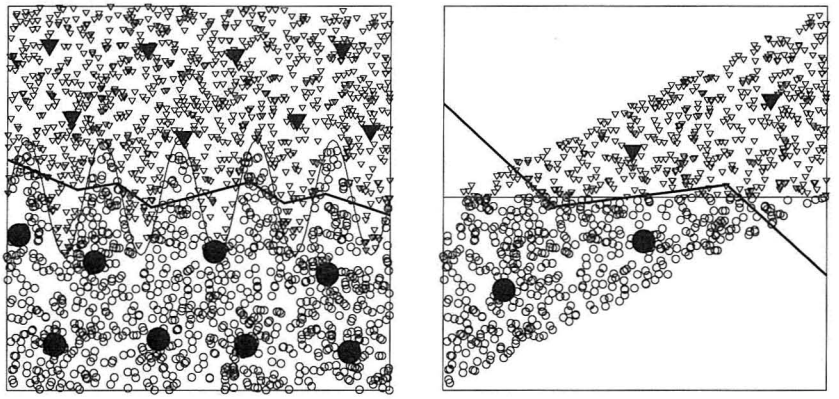
Calling an input and finding the winner among N neurons takes some time Nt_0. The probability that the given neuron will be selected is N^{-1}; if that happens, equation (1) tells us that the neuron will be displaced on average by $\lambda\delta$ down the density gradient. Therefore the mean velocity of a neuron is $\lambda\delta/(N^2 t_0)$. This, multiplied by ϱ, gives the diffusion current $-D\nabla\varrho$. The result is

$$D = \frac{\lambda}{t_0 d A_d^{2/d}} \varrho^{-2/d} N^{-2}. \tag{3}$$

Approximating ϱ by its final mean value N/V and combining equations (2) and (3), we obtain the desired estimate

$$t_{\mathrm{LVQ}} = t_0 \frac{d}{\pi^2 \lambda} N^{2+(2/d)}. \tag{4}$$

The learning time grows with the number of neurons needed for a reliable classification. That number depends on the nature of the problem. As shown below, in some typical cases the idea of habituation may offer a better chance at good performance than simply increasing the number of neurons.



(*a*) (*b*)

Figure 1: Hard cases for Learning Vector Quantization: (*a*) rugged decision surface, (*b*) asymmetric data set. Two non-overlapping classes of the training set (small triangles and circles) are represented by the respective classes of neurons (big triangles and circles). The boundary of neuronal classification (thick line) fails to resolve details of the true decision boundary (thin line). The classification performance is 91.2% in case (*a*), 94.7% in case (*b*).

3. Habituation in LVQ

In LVQ the simplest feature calling for habituation is this (figure 1(*a*)): neurons are more or less uniformly distributed in the respective ranges of the input space, therefore they fail to give a close resolution of a rugged decision surface with many details. To avoid poor classification a very large number of neurons is needed, which makes the learning process slow.

Bringing neurons closer to the decision surface is advantageous in other situations as well. If data sets to be classified are asymmetric in shape of the occupied region (figure 1(*b*)) or in volume then LVQ gives a systematic classification error. You can get rid of that by using a very large number of neurons, or as we suggest by habituation.

Evidently neurons a long way from the decision surface are useless; however, in LVQ they remain there as a consequence of the many training patterns a long way from that surface. Those patterns, once recognized, carry no more useful information for the decision process. Our aim is to get habituated to these boring inputs and get rid of them in further learning.

All we need is a good criterion to determine which are the unimportant input patterns. Our suggestion is this:

1. If input pattern $(\mathbf{x}, q)$ finds no neuron of a different class ($S(e, q) = -1$) within distance $\mathcal{R}(t)$ then it is unimportant and it should be canceled from further learning, with some probability p.

2. $\mathcal{R}(t)$ should be chosen initially about the diameter of the input range, then shrunk in time as learning proceeds.

The sensitive point of the procedure is to find a good shrinking schedule $\mathcal{R}(t)$. That will be done in the next section.

The expected final state is one from which all input patterns except those close to the decision surface have disappeared, neurons being dragged by the remaining ones into the relevant region. This gives an enhanced resolution of fine details in the decision surface, that is, better classification performance with the same number of neurons or equal performance with less neurons. Section 5 presents illustrative results of simulations confirming that expectation.

One might think that it is much easier to test input patterns by their distances from patterns, not neurons, of the opposite classes. That is true for compact, noiseless, non-overlapping training-set classes. Our choice makes the procedure work for the overlapping case too: it starts as LVQ proper and separates neuron classes, which can subsequently guide habituation.

4. Basic estimates

LVQ can achieve arbitrarily good classification by increasing the number of neurons, which costs computing time both in learning and in carrying out the learned classification task. The main advantage of habituation is that it can achieve the same performance using fewer neurons. The price for it is a more lengthy learning procedure; however, as shown below in detail, the reduction in the number of neurons can nevertheless result in faster learning if the classification task is complex enough. The speed-up in carrying out the learned classification is obtained in all cases, since no trace has to be kept of neurons irrelevant to the classification task.

The basic factor limiting the speed of habituation is the possibility of losing neurons if the shrinking of $\mathcal{R}(t)$ is too fast: a neuron lagging too much behind the moving boundary of the active input range will no longer be chosen a winner. The optimal shrinking schedule is the fastest that still avoids that error.

Below we restrict ourselves to $p = 1$; that is, each "unimportant" pattern is erased.

For a quantitative estimate let us assume that we have a data set of two non-overlapping classes, filling out uniformly the two halves of a d-dimensional cylindrical region of d-dimensional volume $2V$ and $d-1$-dimensional cross-sectional area C.

In order to avoid neuron losses, our best chance is to keep the lagging ones just pinned onto the moving boundary of the still-active input range. Indeed, there they feel the whole pulling force of a hemisphere-like Voronoi cell. If immersed more into the active range, some of the input would pull them backwards; if lagging behind, less than a hemisphere chooses them a winner.

The requirement is that the drift of a neuron on the moving boundary should not be slower than the displacement of the boundary itself. The drift velocity can be calculated along the lines of section 2. It is a simple exercise to show that the center of mass of the Voronoi hemisphere of radius R_{cell} is ahead of the center (the neuron) by

$$\delta = \frac{A_{d-1}}{(d+1)A_d} R_{\text{cell}}. \tag{5}$$

Then the equality of drift velocity $\lambda\delta/(2N^2t_0)$ and boundary velocity gives an equation connecting the habituation radius $\mathcal{R}$ to the Voronoi cell radius R_{cell}:

$$\alpha R_{\text{cell}} = -\dot{\mathcal{R}} + \dot{R}_{\text{cell}}, \tag{6}$$

where the dot denotes the time derivative and $\alpha = \lambda A_{d-1}/(2(d+1)N^2t_0A_d)$ (the 2 appears in the denominator because a hemisphere gives a half chance of being chosen a winner), and the last term on the right-hand side accounts for the fact that the first neuron of opposite category is found at a distance R_{cell} beyond the decision boundary.

A second equation connecting $\mathcal{R}$ and R_{cell} is obtained by expressing R_{cell} through the neuron density $\varrho = (A_d R_{\text{cell}}^d)^{-1}$, which is in turn determined by diffusion behind the moving boundary. For an exploratory study we can start from the simplifying assumption that diffusion is fast enough to sustain a uniform neuron distribution in the shrinking active range. Then the average density of neurons is $\varrho = N/((\mathcal{R}(t) - R_{\text{cell}})C)$. Comparing the two expressions for ϱ, we obtain

$$\mathcal{R} = \beta R_{\text{cell}}^d + R_{\text{cell}}, \tag{7}$$

where $\beta = NA_d/C$.

$\mathcal{R}$ can be eliminated from equations (6) and (7) to obtain a single differential equation for the Voronoi radius $R_{\text{cell}}(t)$, with the solution

$$R_{\text{cell}} = [\gamma(t^* - t)]^{1/(d-1)}, \tag{8}$$

with $\gamma = \alpha(d-1)/(\beta d)$. The integration constant t^* is determined by the initial value R_{V0}. Finally the desired shrinking schedule $\mathcal{R}(t)$ is obtained by substituting equation (8) into equation (7). For a numerical evaluation we need $A_d = \pi^{d/2}[(d/2)\Gamma(d/2)]^{-1}$.

t^* is an upper bound for the time needed for habituation (actually habituation has to be stopped somewhat earlier, see below in section 5). Using $NA_dR_{V0}^d = V$, from equation (8) with the definition of γ we obtain

$$t_{\text{habit}} = t_0 \frac{2d(d+1)}{\lambda(d-1)} \frac{A_d^{1+(1/d)}}{A_{d-1}} \frac{V^{1-(1/d)}}{C} N^{2+(1/d)} \tag{9}$$

where the factor $V^{1-(1/d)}/C$ measures the oblongness of the input region.

Equation (9) should be compared to the corresponding result for LVQ without habituation, equation (4), taking into account that habituation allows the use of fewer neurons for the same task. Let us consider the case when the two input classes are separated by a rugged decision surface on which we want to resolve details of a characteristic wavelength l (figure $1(a)$) using $2N$ neurons, with N neurons representing each class.

For the desired resolution the average distance between neurons should not be longer than l. Since, in LVQ, N neurons fill a d-dimensional volume V whereas in the final state of habituation they fill a $d-1$-dimensional area C, the number of neurons needed to represent each class is

$$N_{\mathrm{LVQ}} = V\, l^{-d}/A_d \tag{10}$$

in the first case and only

$$N_{\mathrm{habit}} = C\, l^{-(d-1)}/A_{d-1} \tag{11}$$

in the second. Substituting into equations (4) and (9), respectively, one obtains

$$t_{\mathrm{LVQ}} = t_0 \frac{d}{\pi^2 \lambda} \frac{1}{A^{2+(2/d)}} \left(\frac{(V)^{1/d}}{l} \right)^{2d+2} \tag{12}$$

and

$$t_{\mathrm{habit}} = t_0 \frac{2d(d+1)}{\lambda(d-1)} \frac{A_d^{1+(1/d)}}{A_{d-1}^{3+(1/d)}} \frac{V^{1-(1/d)}}{C} \left(\frac{C^{1/(d-1)}}{l} \right)^{(2d^2-d-1)/d} \tag{13}$$

The main feature of the above formulas is the very steep rise of computing time needed if you want to resolve finer details using LVQ, and the much less steep rise in the case of habituation. For two dimensions you have $t_{\mathrm{LVQ}} \propto l^{-6}$ whereas $t_{\mathrm{habit}} \propto l^{-5/2}$. That proves our claim that habituation, although slower than the original LVQ if applied to simple tasks, may be faster in learning harder ones.

5. Simulations

Although, as stated at the end of section 3, habituation can deal with noisy, overlapping training set classes too, our illustrative examples contain no overlapping classes. Let us begin with some remarks on the character of know-how for potential users, formalizing our experience with the method.

The first of them is this: the random process may sweep several neurons into one pocket of the rugged decision surface, leaving other parts poorly represented (physicists may call this a non-ergodicity effect). To assure good wetting of the decision surface by the available neurons, you have to keep λ large during the whole process of habituation, and start diminishing ("cooling") it only afterwards.

The second remark specifies what "afterwards" means. Shrinking $\mathcal{R}(t)$ should be stopped somewhat before t^*: as soon as the neurons get as close to

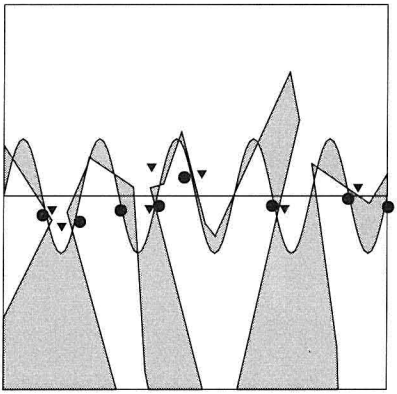


Figure 2: The overfitting error caused by neurons getting too close to the decision surface. The hatched regions are misclassified.

the decision surface as they are to each other, that is, when $NA_{d-1}R_{\text{cell}}^{d-1} = C$ is reached. With equations (7) and (8) this can be rephrased by requiring that shrinking stop as

$$\mathcal{R}(t_{\max}) = \mathcal{R}_{\min} = l\left(\frac{A_d}{A_{d-1}} - 1\right)\left(\frac{N_{\text{habit}}}{N}\right)^{1/(d-1)} \tag{14}$$

is passed. Otherwise a kind of overfitting error develops, since the orientations of local decision surface segments become highly sensitive to small lateral displacements of neurons (figure 2).

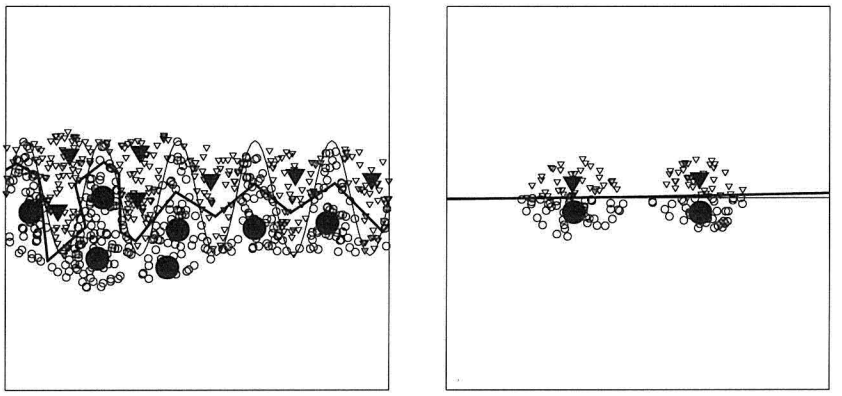
It is slightly paradoxical that the density n of the training set drops out from our formulas for the learning time. It is a straightforward consequence of Kohonen's rule equation (1) that the mean displacement of a neuron per one call of a training pattern depends only on the mean distance between neurons, not on that of patterns. However, if you have a chance to generate more examples of the rule, that improves the classification obtained in smoothness and the accuracy of generalization.

In a computer program it may be advantageous to use an *epoch*, that is, a cycle in which each element of the active training set is called just once, as the unit of time. An epoch lasts $t_E = nC(\mathcal{R} - R_{\text{cell}})Nt_0$ units of real time, thus it gets shorter and shorter as habituation proceeds. It is easy to see that, on this shrinking scale of time, equation (8) is replaced by

$$R_{\text{cell}} = \frac{R_0}{1 + \tilde{t}/\tau}, \tag{15}$$

where $\tilde{t} = t/t_E$ is the time measured in epochs, and

$$\tau = \frac{2d(d+1)A_d/A_{d-1}}{\lambda n V}\,\frac{V^{1-(1/d)}}{C}\,N^{1+(1/d)}. \tag{16}$$



(a) (b)

Figure 3: Learning Vector Quantization improved by habituation. The same neurons as in figure 1, if brought closer to the decision boundary by gradually omitting unimportant training set elements, give a more accurate representation of the boundary. The classification performance grew to 93.9% in case (a) and to 98.5% in case (b).

The characteristic time t^* has been transformed into $\tilde{t}^* = \infty$.

Again, the suggested shrinking schedule is contained in equation (7), this time combined with equation (15). Nothing changes with equation (14) specifying the value of $\mathcal{R}(\tilde{t})$ where habituation should be stopped.

The final states of some illustrative runs are presented in figures 3(a) and 3(b), corresponding to their respective LVQ counterparts, figures 1(a) and 1(b). Figure 4 illustrates our claim that habituation may help one reach good classification with very few neurons in hard cases, whereas LVQ would need a huge number of them.

6. Outlook

Tests on real-life problems like those mentioned in the introduction remain to be done. The performance would depend very much on the nature of the problem. In our first simulations on simple model tasks habituation did not seem to offer considerable savings in learning time for the rugged decision surface case (figures 1(a) and 3(a)), whereas it proved very helpful in learning the asymmetric learning set case (figures 1(b) and 3(b)).

In any case, habituation can considerably reduce the time needed to carry out the learned classification because of the smaller number of neurons needed for equal performance. Although that time is very short compared to that of learning, in various applications to fast on-line data processing its reduction can be an important advantage. Recognizing pre-defined classes of events at high-energy particle accelerators, mentioned in the introduction, is one of

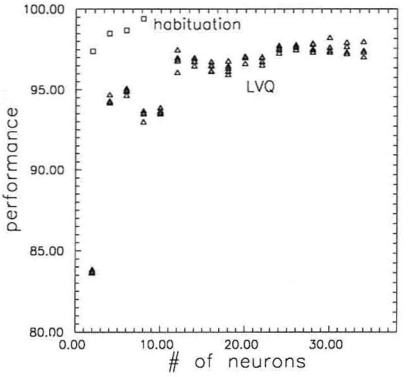


Figure 4: Classification performance versus number of neurons for the case of asymmetrical data sets: LVQ without habituation (figure 1(b)) and habituation (figure 3(b)).

the real applications envisaged.

Finer details of Kohonen's LVQ model can be given a simple physical interpretation in terms of transport of neurons under the driving forces of two concentration gradients: their own (diffusional transport) and that of the training set (analogous to electric conduction). The two transport coefficients are related through an Einstein-type relation, determining the character of the final state of learning and offering an explanation for the sharpness of classification obtained by LVQ in noisy cases. These aspects of the problem are described in [16]. Habituation makes the "electric" driving force time-dependent, controlled by the externally imposed shrinking schedule and the feedback from neuron motion.

Our approach can also be regarded as a kind of genetic algorithm in which the active training set evolves so as to bring out the features most significant to the classification task, in the spirit of reference [11]. Parallel to that, however, the learning process is also progressing; therefore our algorithm is actually based on the co-evolution of patterns and neurons, as pointed out by Wong [17]. Habituation introduces a feedback making the training set distribution and therefore the quasi-electric driving field time dependent. LVQ and habituation as evolutionary models will be treated in a subsequent publication.

Although the idea of habituation is presented here for LVQ, we do not think that the idea is restricted to that particular learning algorithm; thus we are still considering other possible implementations.

Acknowledgments

T. G. is indebted to John Hertz, who first drew his attention to principal component analysis, which motivated this work; to Pál Ruján and Wolfgang Kinzel for discussions on their approach; and to K. Y. M. Wong for suggesting

the co-evolution analogy. This work was supported partly by the Hungarian Academy of Sciences Foundation 'OTKA,' No. I/3. 2179, and by the New Wave Foundation.

References

[1] E.R. Kandel, "Small Systems of Neurons," *Scientific American*, **241** (1979) 60–70.

[2] *Brain Organization and Memory*, edited by J. L. McGaugh, N. M. Weinberger, and G. Lynch (New York, Oxford University Press, 1990).

[3] E. Oja, "A Simplified Neuron Model as a Principal Component Analyzer," *Journal of Mathematical Biology*, **15** (1982) 267–273.

[4] E. Oja, "Neural Networks, Principal Components, and Subspaces," *International Journal of Neural Systems*, **1** (1989) 61–68.

[5] A. Krogh and J. A. Hertz, "Hebbian Learning of Principal Components," pages 183–186 in *Parallel Processing in Neural Systems and Computers*, edited by R. Eckmiller, G. Hartmann, and G. Hauske (Amsterdam, North-Holland, 1989).

[6] J. Rubner and P. Tavan, "A Self-Organizing Network for Principal-Component Analysis," *Europhysics Letters*, **10** (1989) 693–698.

[7] P. Baldi and K. Hornik, "Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima," *Neural Networks*, **2** (1989) 53–58.

[8] J. A. Hertz, "Statistical Dynamics of Learning," pages 137–153 in *Statistical Mechanics of Neural Networks*, edited by L. Garrido, *Lecture Notes in Physics*, **368** (Berlin, Springer, 1990).

[9] W. Kinzel and P. Ruján, "Improving a Network Generalization Ability by Selecting Examples," *Europhysics Letters*, **13** (1990) 473–477.

[10] T. L. H. Watkin and A. Rau, "Selecting Examples for Perceptrons," *Journal of Physics A*, **25** (1992) 113–121.

[11] N. H. Packard, "A Genetic Learning Algorithm for the Analysis of Complex Data," *Complex Systems*, **4** (1990) 543–572.

[12] I. Csabai, F. Czakó, and Z. Fodor, "Quark- and Gluon-jet Separation Using Neural Networks," *Physical Review D*, **44** (1991) R1905–R1908.

[13] I. Ladunga, F. Czakó, I. Csabai, and T. Geszti, "Improving Signal Peptide Prediction Accuracy by Simulated Neural Network," *Computer Applications in the Biosciences*, **7** (1991) 485–487.

[14] T. Kohonen, *Self-Organization and Associative Memory*, 2nd edition (Berlin, Springer, 1988), page 199.

[15] T. Kohonen, "An Introduction to Neural Computing," *Neural Networks*, **1** (1988) 3–16.

[16] T. Geszti, "Hydrodynamics of Learning Vector Quantization," to appear in *From Phase Transitions to Chaos*, edited by G. Györgyi, I. Kondor, L. Sasvári, and T. Tél (Singapore, World Scientific, 1992).

[17] K. Y. M. Wong, private communication (1991).