Evolutionary learning of small networks

Thomas Filk$^{1,2,3}$ and Albrecht von Müller$^{1,4}$

$^1$ Parmenides Center for the Study of Thinking, Kardinal-Faulhaber-Str. 14a, 80333 Munich, Germany

$^2$ Department of Physics, University of Freiburg, Hermann-Herder-Str. 3, 79104 Freiburg, Germany

$^3$ Institute for Frontier Areas in Psychology and Mental Health, Wilhelmstr. 3a, 79098 Freiburg, Germany

$^4$ Department of Philosophy, University of Munich, Geschwister-Scholl-Platz 1, 80539 Munich, Germany

email: thomas.filk@t-online.de
albrecht.von.mueller@parmenides-foundation.org

Abstract

Results are presented of a simulation which mimics an evolutionary learning process for small networks. Special features of these networks include a high recurrence, a transition function which decreases for large input activities, and the absence of tunable weights attached to the lines - the line is either there (weight 1) or absent (weight 0). It is remarkable that already these simple systems exhibit a complex learning behavior and the phenomenon of punctuated equilibrium in the evolutionary process. These findings should be of interest for both, the general understanding of evolutionary dynamics and, more specifically, the understanding of the role of recurrence in combination with non-monotic response patterns for learning processes.
1 Introduction

Viewed from a simplified perspective, many systems can be modelled as consisting of partially connected units. In a general context, the connections represent the possible transport routes of energy, data, or information, and the units serve as processing modules. Connections can be directed or undirected, depending on whether the flow of energy or data proceeds only in one direction or both ways. The output of each unit depends on the input as well as internal features defined for this unit. Such a structure is usually called a “network”, or more specifically a “neural network”. The investigation of complex networks and neural networks has become a major tool in different areas of science (for networks in general, see e.g. Bornholdt and Schuster [1] and references therein, a comprehensive introduction to the theory and applications of neural networks can be found in Haykin [2]).

Mathematically, the units and the connections in a network are represented by graphs - a set of vertices and a set of (directed or undirected) lines connecting these vertices. “Energy”, “data” or “information” are represented by a field - we will use the term “activity” - on the set of vertices for which a time evolution is defined. In addition to the dynamics of the activity, the evolutionary-like process includes a stochastic algorithm for the insertion or deletion of lines. This algorithm represents the learning process of the graph. In the following, the term “graph” will be used for a set of vertices together with the set of lines connecting these vertices, while the term “network” refers to the graph plus the dynamics of the fields defined on the vertices.

A particular interesting field of applications is the architecture and dynamics of brains on the fundamental level, where the units correspond to neurons and the lines to synaptical connections. Since more than 30 years, neural networks constitute a major tool in computational biology. In this case, the activity field represents the neural firing rate and the connections the synaptical weights. Note, however, that we do not include explicit weights to the lines representing a “transmissivity” for the signal. A line is either present (weight 1) or absent (weight 0).

The model presented here mimics an evolutionary learning process. Imagine an organism which has to survive in a given unchanging environment. The organism has only a limited perception of the environment and it has even less possibilities to react to these perceptions. The signals from its environment enter the organism via a set of peripheral receptors and the reaction of the organism is triggered through firing rates at certain effectors. Between receptors and effectors is the internal neural system which processes the incoming neural activity and generates the outgoing neural activity. Suppose this organism is based on only one type of neurons and only one type of synaptic connections. The only changes in the organism which can improve its reactions to certain perceptions are changes in the way the neurons are connected among each other.

In the model we investigated, the network can “perceive” eleven different external input patterns or “images”, which are grouped into three different sets. For each set an optimal output reaction is predefined. These three different optimal reactions correspond to particular activities at “output”-vertices.
The evolutionary process consists of random mutations which affect the “synaptical connections”, i.e., the way the vertices are connected by (directed or undirected) lines: a connection which was present in the previous generation can be absent in the “offspring”, or a certain synaptical connection can suddenly be present in the offspring although it was not present in the parent generation. In this respect the simulations mimick so-called “direct genetic encoding”, i.e., the mutations change single synaptical connections (as if these are determined by the genetic code) but not the algorithm by which general features of the network are generated. Only the fitness of the organism determines which species will survive. The fitness is measured by the way the organism reacts to what it perceives from its environment compared to the optimal reaction possible. If no single mutation can improve the fitness of an organism, the process stops. We investigated several thousand of these processes, and we found a few hundred networks which learned to react optimally to the input patterns.

Compared to most of the neural networks described in the literature (see Haykin [2] and references therein), the models presented here differ in several aspects. First of all, the relevance of directed compared to undirected lines is investigated. We made simulations for undirected graphs and for directed graphs, where even for directed graphs the initial random graph was chosen to be undirected. Undirectedness leads to a maximal recurrency, feedback, or self-reference, which we think to be of major importance in real organisms. Even when the simulation procedure allows for directed lines, it turns out that a certain percentage of lines remains undirected (or, equivalently, there are directed lines in both directions). Secondly, “learning” improves only through “natural selection”, i.e., the simulations do not make use of any explicit back reaction of the environment onto the organism like, e.g., backward propagation algorithms. Furthermore, the transmissivity of the synaptic connections is either 1 or 0 (the line is there or it is absent); there are no weights attached to the lines.

Finally, the dynamics by which a vertex responds to the incoming “firing rate” is such, that the outgoing firing rate of a vertex increases with the integrated sum of the incoming firing rates up to a certain maximal input value. If, however, the incoming firing rates exceed this maximal input value, the outgoing firing rate of a vertex decreases again and can even become zero, if the input firing rates are too large. This idea was motivated by different investigations of the logistic mapping for coupled map lattices (see, e.g., [3, 4] and references therein), as well as theoretical models for neurons [5]. Recent experimental findings by Destexhe [6] and Léger et al. [7] suggest that this behavior may indeed be realized for certain types of neurons.

In the next section, we describe the model and the simulation procedure in more detail. Section 3 contains the results of our simulations concerning the evolutionary process. Most important is the phenomenon of “punctuated equilibrium” in the process. In the context of evolutionary processes this expression was introduced by Eldredge and Gould in 1972 [8]. It has been reported to occur in simulations of evolutionary adaptation processes by Fontana and Schuster [9] and by Crutchfield and van Nimwegen [10, 11]. (Schuster and Fontana speak of “major transitions”, and Crutchfield and van Nimwegen refer to the phenomenon as the “royal staircase”.) Punctuated equilibrium has also been described
in the context of learning procedures in organizational systems (see Price and Evans [12] and Gersick [13]). However, a complete understanding of this phenomenon in the context of learning processes seems to be still missing.

Apart from the feature of punctuated equilibrium we investigated the relevance of image dependence (modularity) for this phenomenon and the role of pruning. In Section 4, we describe the properties of the graphs which became optimal learners, the sequential learning of the graphs and the impact of “lesions” (random deletion of lines or vertices). The conclusion discusses possible extensions of this project as well as possible impacts of these results outside the field of dynamical systems.

2 Description of the Model and the Simulation Procedure

The mathematical systems which we investigated in numerical simulations consist of graphs together with a dynamics of fields defined on the vertices of these graphs. A (simple) graph is defined to be a set of vertices (or points) \( V \) together with a relation “\( \rightarrow \)” between vertices, where \( y \rightarrow x \) denotes a directed line from vertex \( y \) to vertex \( x \). If two points are connected in both ways, i.e., both relations \( y \rightarrow x \) and \( x \rightarrow y \) are present, the line between \( x \) and \( y \) is called undirected. If all lines are undirected, the graph is called undirected. The restriction to “simple” graphs means that there are no self-loops (no vertex is connected with itself) and no multiple connections for one direction (for each pair \( (x, y) \) there exists at most one line from \( x \) to \( y \) and from \( y \) to \( x \)). The connections (relations) of such a graph are described by the adjacency matrix:

\[
A(x, y) = \begin{cases} 
1 & \text{if there is a directed line from } y \text{ to } x \\
0 & \text{else}
\end{cases}
\] (1)

We define a certain set of vertices to be input vertices (“receptors”) and a second set of different vertices as output vertices (“effectors”). Input and output vertices together are referred to as peripheral vertices. All other vertices belong to the set of “internal vertices” (sometimes called hidden vertices). We require that there are no directed lines entering the input vertices (such lines would have no effect, because the activities of the input vertices are determined by the input patterns and are not updated by the dynamics of the system), and there should be no lines connecting input and output vertices directly. All simulations used 16 input vertices and 2 output vertices. The number of internal vertices differed: for undirected graphs we performed simulations with 14 and 32 internal vertices, for directed graphs we made simulations with very small graphs - 4 and 6 internal vertices - and for larger graphs with 57 internal vertices.

On the set of vertices we define a field (called “activity” or “firing-rate”) \( u(t, x) \): at each “time step” \( t \), the activity at each vertex \( x \) can assume one of several values: \( u(t, x) \in \{0, 1, 2, \ldots, u_{\max}\} \). The configuration \( \{u(t, x)\}_{x \in V} \) will be called the state of the system at time \( t \). The restriction to discrete (integer) values has purely programming
reasons. The dynamics is defined by the following equation:

\[ u(t + 1, x) = f \left( \sum_{y \to x} u(t, y) \right) = f \left( \sum_y A(x, y) u(t, y) \right), \]  

(2)

where \( f() \) is the transfer function (see below) and \( \sum_{y \to x} \) implies a summation over all vertices \( y \) for which a (directed) line from \( y \) to \( x \) exists. Hence, the activity \( u(t + 1, x) \) of vertex \( x \) at time \( t + 1 \) depends only on the sum of the activities of the neighbors \( y \) at time \( t \). The transfer function \( f(x) \) (see Fig. 1) is a triangular function defined by:

\[
f(x) = \begin{cases} 
\frac{u_{\text{max}}}{u_0} \cdot x & \text{for } x \leq u_0 \\
\frac{u_{\text{max}}}{u_1 - u_0} \cdot (u_1 - x) & \text{for } u_0 < x < u_1 \\
0 & \text{for } x \geq u_1
\end{cases},
\]

(3)

where \( |u| \) denotes the smallest integer larger or equal to \( u \).

Figure 1: The transfer function \( f(x) \). The simulations used the values \( u_{\text{max}} = 10 \) and \( u_1 = 30 \). The values for \( u_0 \) varied between \( u_0 = 8 \) and \( u_0 = 15 \). Most simulations were made with \( u_0 = 10 \).

While the usual choices for the transfer function exhibit a monotonic behavior - either as a discrete step function or as a smeared out S-shape function - we decided to choose a transfer function which drops off for large input values. As already mentioned in the introduction, this choice was motivated by theoretical models for neurons [5] and recent experimental findings [6, 7]. Another reason was that non-monotonic transfer functions (in particular the logistic map) have shown interesting behavior in the context of coupled map lattices (see, e.g., [3, 4]) which can be interpreted as a neural network on regular lattices. The drop-off for large input values not only prevents an “explosion” of activity but also allows for inhibitory effects: once the input activity exceeds \( u_0 \) any further input leads to a reduced output.

For the input vertices the value of the activity was set to 0 or \( u_{\text{max}} = 10 \) according to a set of “input images” or input patterns which were “presented” to the network. Figure 2 shows the 11 input patterns used in the simulations. The input patterns are grouped into three different sets (suggestively referred to as “close danger”, “distant danger”, and “food”). The first set contains only image 1, the second set consists of images 2, 3, 4, and
Figure 2: The 11 input patterns, shown as $4 \times 4$ matrices, which were presented to the 16 input vertices. ◦ indicates activity 0, • indicates activity $u_{\text{max}}$.

5, and the other six images 6–11 belong to the third set. Note that for the images in group two and three the number of vertices with activity 0 and maximal activity is the same: the system cannot distinguish these patterns by simply counting the number of vertices with maximal activity. Furthermore, it should be obvious that neither the arrangement of these vertices in form of a $4 \times 4$ matrix nor the "interpretations" of these images in terms of danger or food are of any relevance or meaning to the system. (Equivalently, one may think of the input-vertices as olfactory receptors.)

To each group of input patterns we define an optimal reaction of the system. The reaction is the activity of the two output vertices. The optimal reactions for the three different groups are: $(0, 0)$ for group 1 (sometimes referred to as "don’t move"), $(10, 0)$ for group 2 ("run away"), and $(0, 10)$ for group 3 ("eat"). Again, the interpretation of the optimal reactions has no relevance for the system. An “optimal learner” is a network for which the activities at the output vertices agree, after a few transitory time steps, with the predefined optimal reaction for each of the 11 images. (Later we will distinguish “optimal learners” for which this is true if the 11 images are presented in a fixed order, and “perfect learners” for which this is true independent of the order of the presentation of the images or the initial conditions of the activities.) Our aim is to find such graphs and to investigate their properties.

The following expression defines a measure for the performance of a network. It may be interpreted as an inverse fitness of the system:

$$\Delta = \sum_{\text{11 images}} \sum_{\text{20 time steps}} \sum_{\text{2 output vertices}} (u(t, x) - u_0)^2. \quad (4)$$

The sum extends over the 11 input patterns, 20 time steps according to the defined dynamics, and the two output vertices. In each case, the actual activity at the output vertices is measured, the optimal reaction as defined above subtracted, and the square taken in order to obtain a positive number. This measure will be called the “performance function” of the system with respect to the optimal reaction. $\Delta$ vanishes for “optimal learners”, while for random graphs it assumes values around 15000 to 25000. After the input pattern has changed the system is allowed to “relax” for a certain number of transient time steps (usually 10–20 time steps). After this relaxation time has passed the output is
measured. In order to analyze image dependent modularity structures, we also measured
the performance function for single input patterns, i.e., there is no summation over the
11 input images. To avoid confusion as to which performance function we refer to, we
sometimes distinguish the “total performance” from the “single image performance”.

The simulation procedure for obtaining optimal learners mimics an evolutionary sce-
nario: starting from a random adjacency matrix (an arbitrary “species”), we obtain a
sequence of new graphs (new “species”) by randomly making small changes (“mutations”)
in the connectivities of the graph and measuring the performance function (inverse “fit-
ness”) of the new graphs. If the performance function is smaller than for any previous
graph, i.e., if the fitness is better, the new graph is accepted and becomes the “surviving
species”, otherwise it is rejected and the “fitter” graph survives. In order to distinguish
between the single time steps of the iterative dynamics (Eq. 2) and the iterative sequence
of graphs obtained by inserting and deleting lines, we refer to the steps of the second one
as a “generation step”. One generation step comprises 330 or 440 time steps, depending
on whether the relaxation time (transitory time) after each image change is chosen to be
10 or 20 time steps.

In detail, the simulation procedure is as follows: We first define a random adjacency
matrix for a given number of vertices by setting the entries in the matrix to 1 or 0 with a
fixed probability. This probability depends on the number of vertices and varies between
2% (for the largest graphs with a total of 75 vertices) up to 40% for the smallest graphs
with 22 and 24 vertices. This matrix is symmetrized, i.e., it becomes the adjacency matrix
for an undirected random graph. All entries which correspond to a connection between
an input and an output vertex are set to 0. Furthermore, the entries in the adjacency
matrix corresponding to a connection from internal vertices to input-vertices are set to
zero. Only with respect to these lines the initial random graph is directed.

Each image is presented to the graph for a total of 30 or 40 time steps, and for the
last 20 time steps the difference between the actual activity and the predefined optimal
activity (the “error”) is measured. The images are always presented in the same sequence
(not in randomized orders). In a next step, one of the connections is changed by inserting
or deleting a single line for two randomly chosen vertices which are not both input and/or
output points. Depending on whether the simulation is restricted to undirected graphs
only or allows for directed lines, the inserted or deleted line can be directed or undirected.
Again the performance is measured by presenting the images and if the performance
function is smaller than for the previous graph, the line change is accepted (the new graph
becomes the “surviving species”), otherwise it is rejected (the new graph “dies out”). This
procedure is iterated until a certain number of successive line changes (between 1500 and
6000 generation steps, depending on the size of the graph) has been rejected, in which
case the run stops.
3  The evolutionary process

In most cases, the process stops at a graph for which the performance function is considerably smaller than 10000, although in a few cases the graphs did not improve below a performance of about 12000. For undirected graphs, about 3% of all runs lead to a graph with a performance function smaller than 100 and about 0.15% of all runs produced an “optimal learner” with a performance function of $\Delta = 0$. For directed graphs the percentage of good learners with a performance function below 100 was about 5% and in roughly 2–3% of all cases the process ended with an optimal learner with error function $\Delta = 0$.

3.1 Punctuated equilibrium

Figure 3: Two examples of evolutionary processes. The performance function is presented as a function of “generations”. Each “generation” corresponds to a new graph which is obtained from the previous graph by a random line change. At each step only the graph with the best performance “survives”. Both examples correspond to directed graphs with 48 vertices.

The most obvious and interesting feature of the evolutionary process is the phenomenon of “punctuated equilibrium”. The performance function does not decrease gradually, but rather exhibits periods of intense tumbling (transitions) intermitted by periods of almost no changes in variance at all. Figure 3 shows two examples.

While the error function for the individual processes exhibits an obvious step-like behavior, the average over a sample of 1000 optimal learners (size $n = 48$) shows a smooth decrease (see Fig. 4a). The logarithmic plot of the same graph (Fig. 4b) reveals a rapid decrease in the beginning (less than 2000 generations) and a slow approach to an almost exponential decrease. The line drawn in Fig. 4(right) corresponds to a “half-life” of about 3200 generations for the exponential fit to the curve in the regime where the number of generations is greater than about 4000.

For a better statistical investigation of this phenomenon we determined the frequency $P(L)$ of the stopping times for the evolutions processes for a sample of 1000 optimal learners. Let $L$ be the number of generations for which the fitness remains constant (no rewiring of the vertices has been accepted), then $P(L)$ is the frequency of how often
Figure 4: The learning curve averaged over a sample of 1000 evolutionary processes for optimal learners of size \( n = 48 \). (a) The performance (error function) as a function of generations; (b) logarithmic plot of the same function.

Figure 5: (a) Logarithmic and (b) double logarithmic plot of the distribution of stopping times \( P(L) \) for a sample of of 1000 evolutionary processes for optimal learners of size \( n = 48 \). (The stopping times were grouped in bins of size 10.) The solid line in the right plot corresponds to a power law with exponent \( k \approx 2 \).

The occurrence of punctuated equilibria is surprising because the transition phases are not induced by changes in the environment (this could have been simulated by a redefinition of the optimal outputs during an evolutionary process), but triggered by internal changes. Similar features have also been observed in simulations of RNA folding [9] and
other evolutionary systems [10, 11]. In the present context we use the expression “punctuated equilibrium”, because the general structures of the “organism” remain the same. Minor changes in the network connectivity lead to a large increase in fitness. Punctuated equilibria seem to be a characteristic feature of most evolution-like processes provided the dimension of the configuration space is large, the evolutionary task is complicated enough, and the training process is based on trial and error.

If the evolutionary process is interpreted as a random walk in a high dimensional fitness landscape, the periods of constant performance function correspond to local “almost minima” (metastable states), for which almost all directions lead to an increase in fitness. Only very few directions allow the system to proceed to other parts of the landscape, where most paths lead “downwards”. Note that we do not allow the systems to penetrate a barrier in the fitness landscape. Only if the performance improves is a line change accepted.

There might also be a connection to so-called “neutrality systems” for which most directions in the fitness landscape do not change the fitness. During the long periods of constant fitness the systems “explores” the neutrality plane until it finds an escape to regions with a lower fitness.

3.2 The modularity of the transitions

The occurrence of transitions between the equilibria raises the question whether the rapid decrease of the performance (increase of the fitness) during such a transition is due to a general increase in the performance of the tasks, or due to an increase of the performance with respect to only certain tasks while other tasks are not or almost not involved.

In order to investigate such a modularity or image dependence of the transitions in more detail, we analysed the single image performance during the evolutionary process. Figure 6 shows the results for a single evolutionary process, but the general structures seem to be universal.

Almost all images seem to be involved in the first rapid “tumbling” phase up to step 3000. (These steps now correspond to “generation steps”, not to time steps.) At the end of this phase, images 1, 8, and 9 are almost optimally recognized while the graph performs rather badly with respect to most other images. For the first transition phase between step number 7000 and 10000, a clear increase of the single image error function for image 4 is to be contrasted with almost no changes in the single image error functions for image 5, 8, 9, and 11. Images 2, 6, and 7 are only weakly involved. Images 1 and 3 show even a weak transient increase of the error while image 10 exhibits an even stronger increase during this transition. The second transition phase between step 11000 and step 13000 is induced mainly by image 10, more weakly by images 2, 4, and 11. Images 1, 3, 6, 7, 8, and 9 are not involved at all. Finally, the last transition between steps 14000 and 15000 is mainly due to a decrease of the single image performance function for image 5, while images 2, 4, 7, 8, and 9 are not involved at all. At the end of this transition all images are recognized optimally.

It may be interesting to note that the modularity found in the transitions does not
Figure 6: Modularity in the evolutionary process. The transitions in the total performance function are due to transitions with respect to single images. The upper picture shows the single image performance as a function of iterative steps or “generations”, while the lower picture shows the total performance for the same process. In the upper case only the performance functions for accepted graphs are shown. Note that the “error=0”-line in the upper picture is different for each image in order to avoid all data falling on top of each other. After about 15000 steps the graph becomes an optimal learner and the single image errors are all zero.
seem to be correlated with the output. Although such a statement needs to be confirmed with better statistics, apparently the images involved in a transition are not necessarily related by the same output reaction. This is in contrast to other modularity dependencies which will be mentioned in later sections (see Sections 4.2 and 4.3).

3.3 Pruning

How important is pruning (the deletion of an existing line as compared to the insertion of a line) for the evolutionary process and the prospects to find an optimal learner? This question was one of the initial motivations for studying these models.

Figure 7 shows two examples for learning processes with completely different behavior concerning the importance of pruning. Both figures represent the changes in the number of lines (compared to the number of lines of the initial random graph) as a function of the iterative steps (the “generations”) of the process. An increase, like in Fig. 7(a), indicates a dominance for the insertion of lines, i.e., there is almost no pruning. A decrease (like in Fig. 7(b)) indicates a dominance of pruning, i.e. deletion of lines.

Figure 7: The change in the number of lines relative to the initial random graph during a learning process. The importance of pruning is completely different for these two examples. (a) In this evolutionary process for a (directed) graph with 75 vertices, the almost monotonical increase of the number of lines indicates that pruning plays (almost) no role at all. b) The overall decrease in the total number of lines for this process, obtained for a (directed) graph with 22 vertices, shows a dominance of pruning.

It turns out that the relevance of pruning depends crucially on many parameters, in particular the number of vertices, the line probability for the initial random graph, and the parameters of the dynamics. Therefore, we can not draw any final conclusions concerning the relevance of pruning from the present data. Apart from a more thorough statistical analysis, one of the future projects will be the simulation of processes, where the learning task changes during the process, so the graphs have to “relearn”. We expect pruning to play a more important role in these cases.
4 The structure and the performance of “optimal learners”

This section deals with those graphs which resulted from a successful evolutionary process and which became optimal learners. More important than some of the properties of these graphs is the content of what exactly the graphs have “learnt”.

4.1 The structure of “optimal learner”-graphs

Ever since the work of Watts and Strogart [14], “small world” graphs have been found in almost any discipline where the relations among constituents can be formalized in terms of large networks (see e.g. [15, 16]). The expression “small world” is used for graphs which exhibit the following two properties: (1) The average distance between two arbitrarily chosen vertices is small compared to regular graphs; it increases with the logarithm of the size of the graphs, a behavior which is also found for random graphs. (2) The connectivity between vertices is much larger than for random graphs and more like regular graphs.\(^1\) In general, the “small world” properties are only attributed to undirected graphs, because for directed graphs there exist different definitions of connectivity and mean distance.

A statistical analysis of the two relevant features - average distance and connectivity - for those graphs which show an optimal output response with respect to the input (the “optimal learners”) does not reveal any significant differences as compared to random graphs. This is mostly due to the fact that for graphs with 32–50 vertices (for those sizes we investigated undirected graphs) it is quite difficult to distinguish small world graphs from random graphs. During the evolutionary process, the connectivity shows an apparent increase. However, this is due to an increase in the total number of lines, which implies that the connectivity of a compatible random graph (i.e., a random graph with the same average number of lines) does also increase. Therefore, the apparent increase of small world properties during the evolutionary process may only be superficial.

It can be observed, however, that for most optimal learners there are one or two vertices with an extraordinary large degree as compared to random graphs with the same line occupation probability. This observation is usually associated with a hierarchical structure of the graph, in particular with so-called “scale free” graphs for which the degree distribution for large degrees has a a power law dependence (see, e.g., [16]). For random graphs, the degree distribution is a Poisson function with an exponential decrease for large degrees.

4.2 Sequential learning

As mentioned before, during the simulation of the evolutionary process the images are presented to the graphs in a fixed sequential order, and the vertex field (the neural activity)

\(^1\)In this context “connectivity” is a technical term: the connectivity of a vertex is defined as the relative number of connections among the neighbors of this point compared to the maximal possible number of connections. The connectivity of a graph is the average connectivity of its vertices.
is updated according to Eq. 2. So, there is no randomization of the order of the input patterns nor of the activities during a process. If presented in the same order, an optimal learner recognizes the images perfectly in the sense that the actual output agrees with the predefined optimal output (after the transitory time steps).

We also tried to find “perfect learners” for which the output reaction was optimal independent of the order of the presentation of the images and independent of the initial conditions chosen for the activities (which were randomized before a new image was presented). For networks with only undirected lines, we couldn’t find even one perfect learner in a run consisting of 15000 different evolutionary processes (such a run would have produced roughly 300 optimal learners), and we conclude that this task is considerably more difficult (not just by a factor of the order of the number of images). For networks with directed lines, we found 14 perfect learners in a sample of about 900 optimal learners for graphs with n=48 vertices. For all other optimal learners the “recognition” of the input patterns depended either on the sequential order in which the images were presented and/or on the initial configuration of activities. In Atmanspacher and Filk [17] the dependence of the recognition of patterns on the sequence and the initial conditions was used to define a complexity measure for a learning process.

Figure 8 shows four examples where the presentation of the images to an optimal-learner-graph differs from the presentation during the learning process. While in Fig. 8(a)-(c) the order of the presentation has been changed, Fig. 8(d) shows the performance of the graph when the vertex activity (the “firing-rate”) is randomized before the presentation of each new pattern. One may interpret this randomization as a way to make the graph “forget” any information about the previous image.

Any performance function different from zero means that the graph does not react optimally to the presented picture. The fact that in all four cases there are input patterns which the graph does not recognize optimally implies that the graph has not learned to recognize the 11 images independent of what has happened before. In most cases the graphs learned to recognize only the correct sequence of images. This “sequential learning” can be compared with the sequential learning of words in a foreign language (e.g. in the order they occur in a register) which are only remembered if they are asked in the same order as they have been learned.

In terms of dynamical systems one could say that after the presentation of a new image (which corresponds to a special kind of boundary condition) the graph approaches an image-dependent attractor. This attractor is the initial condition for the dynamical process when a new image is presented, i.e., when the boundary conditions have changed. In general, the attractor which a system approaches for large values of $t$ depends on the initial configuration. “Sequential learning” implies that the system approaches the correct attractor (with an optimal activity of the two output vertices) for a special set of initial conditions which only has to include the correct attractor configuration of the previous image. A “perfect learner”, which recognizes the images independent of their order and independent of the initial conditions, is a much more specific system for which the attractor-basin for the optimal reaction to each image consists of the total configuration space.
Figure 8: Four examples where the presentation of the images is changed compared to the learning process. In each case, the single image error functions for the 11 input patterns are shown. (a) The order of the images within a block is unchanged but the order of the three blocks of images is changed: block 2 is presented before block 1. (b) The images are presented to the graph in reverse order. (c) The images are presented to the graph in random order. (d) The images are shown in normal order but the activities are randomized before a new image is presented.

Figures 8(a)-(c) reveal an “output modularity” in the recognition of the images. While the six images corresponding to the output (0, 10) (“eat”) are mostly recognized, the four images corresponding to the output (10, 0) (“run away”) are not recognized when they are not presented in the correct order. Image 1 is always recognized independent of the previous configuration. This can be attributed to the fact that during the learning process, image 1 is the first image presented to the new graph and, therefore, the recognition of this image is more stable against fluctuations in the initial state of the system than the recognition of the other images. Furthermore, the desired output for image 1 is (0, 0) which is almost the default output for this kind of set-up. (For most input patterns, random graphs yield the output reaction (0, 0).)

We attribute these results to the modularity in the output and not to similar patterns in the input. For the graphs, the 16 input vertices are just 16 randomly chosen vertices, and it would be most surprising, if these simple models would reveal a modularity for the “shape” of the input image. We intend to test this hypothesis in future simulations by
choosing randomly selected input patterns. In this case, any modularity in the behavior of the system is expected to be correlated to the output and not to the input.

4.3 “Lesions” and Stability

What happens to the performance function of a graph, when a line or a vertex is randomly deleted? One expects that the deletion of a line has much less severe consequences for the performance of a graph than the deletion of a vertex, because the deletion of a vertex is accompanied by a deletion of all lines which connect this vertex with other vertices. The main question we wanted to address by deleting single lines or vertices and measuring the single image variance of the resulting graph is again related to modularity: Is an increase of the total error function of the graph after the deletion of a line or a vertex related to particular input or output patterns, or is it due to a general increase of all image dependent variances?

For this purpose we determined the correlation function for the single image performances for the 11 input patterns after deletion of a line or a vertex in a sample of 1000 optimal learners (size n=48). To be more precise: For a given optimal learner we deleted one internal line (an internal line is a line which connects two internal vertices) or one internal vertex, measured the single image performances $\Delta_i$ for the input patterns $i$ and determined the product $\Delta_i \Delta_j$. We repeated this measurement for the same optimal learner but after deletion of a different line (vertex) until all internal lines (vertices) have been tested. Finally we repeated this procedure for 1000 different optimal learners and averaged. The results were the averaged correlation functions $C^L$ and $C^V$ for the deletion of a line and the deletion of a vertex:

$$c^{(L,V)}_{ij} = \langle \Delta_i \Delta_j \rangle.$$

The result is shown in Figure 9.

The modularity dependence with respect to the reaction is obvious for both cases from the block diagonal shape of the correlation matrix. In many cases, only the patterns of one class (“danger”, “distant danger”, or “food”) are not recognized optimally by a network. This is reflected by the fact, that the correlations between different classes are smaller by a factor of 10–100 compared to the correlations within one class. Furthermore, the correlations for different input patterns within one class (“distant danger” and “food”) are smaller by a factor of about 2–5 compared to the diagonal elements. In Fig. 9, the size of the maximal box is scaled to 1. However, the absolute values of the correlations matrixes are smaller by a factor of about 10 for the case where a line has been deleted as compared to the case where a vertex has been deleted. This confirms that the deletion of a line has less severe consequences compared to the deletion of a vertex. Actually, the results indicate that in many cases the deletion of a line has no consequences while the deletion of a vertex almost always has. In a natural selection process the stability of a system against “lesions” would be a further fitness criterion apart from the performance function.
Figure 9: Correlation matrix for single image performances for the 11 input patterns, (a) after the deletion of a line, (b) after the deletion of a vertex. The figures show the average over a sample of 1000 graphs which proved to be optimal learners. Furthermore, for each graph we determined the average with respect to the deletion of an internal line or the deletion of an internal vertex. The linear size of the boxes is proportional to the normalized correlation (for both cases, the largest value corresponds to a box of size 1).

The variance (diagonal element) of the single image performance for input pattern 1 ("large danger") is small compared to the other input patterns. Again, the main reason for this is that the optimal reaction activity at the output vertices for this image is the "default value" (0, 0).

The functional modularity dependence for the deletion of vertices and lines is obvious. However, this should not lead to the conclusion that there are certain "grandmother vertices". Although a particular vertex might be involved in the recognition of a certain class of images (or a certain class of outputs), this vertex is, in general, not the only one which takes part in this process. It turns out that, in general, several vertices are involved in the dynamical reaction to a certain input pattern, while for the other vertices the activity is almost zero. (The average size of these "activated neural assemblies" depends on the size of the graphs and is mostly between 3 and 5 for graphs with 14 internal vertices.)

5 Discussion and Conclusion

The presented paradigm allows for many possible extensions and variations. The following list is far from being complete.

A major future project aims to investigate learning strategies: Which learning strategy improves and accelerates the learning process? Is it better to first learn tasks in sequential order and then randomize, or is it faster to learn the complete task right from the beginning? Should a complex task be split? These are questions which can be investigated in the present framework. Of course, from such results one cannot draw definite conclusions for human learning strategies, but one can get hints which can be further
investigated with psychological methods.

Another project aims to investigate the role of the different parameters of the model: How important is the form of the transfer function \( f(x) \)? How important is the decrease of this function for large input activities? Up to now we investigated only “excitory” neural vertices and synaptical connections. Does the learning process improve significantly if one also includes inhibitory features? Does the process improve if one introduces several types of neurons? How important is pruning, in particular if the tasks are changed during the evolutionary process?

These are but some of the questions which can be addressed by investigating the model described in this article. The model exhibits a remarkable richness, it is simple enough to be investigated in every detail, and at the same time it is complicated enough to yield interesting results.

Acknowledgements

We acknowledge many stimulating discussions with (in alphabetical order) Harald Atmanspacher, Britta Glatzeder, Eduard Kraft, Chris Langton, Michael Öllinger, Ernst Pöppel, Helge Ritter, Stefan Rotter, and Eörs Szathmáry. Furthermore, we would like to express our gratitude to the unknown referee for helpful comments and suggestions to improve the article.

References


