Pattern Recognition Letters 64 (2015) 44–52

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Computation and dissipative dynamical systems in neural networks for classification ☆

Frank van der Velde [a,b,*]

[a] University of Twente, CPE-BMS, Drienerlolaan 5, 7522NB Enschede, The Netherlands
[b] IO, Leiden University, The Netherlands

ARTICLE INFO

Article history:
Available online 2 March 2015

Keywords:
Classification
Classical cognitive science vs. dynamical systems
Computation
Dissipative systems
Induction
Neural networks

ABSTRACT

Foundational issues related to learning, processing and representation underlying pattern recognition have been discussed in history and in recent times. The scientific approach to pattern recognition could provide new tools to investigate these foundational issues, which in turn could inform the scientific approach to pattern recognition as well. One such tool could be the analysis of learning, processing and representation in connectionist (or neural) networks, which have been extensively used for pattern recognition. Based on a mathematical analysis of the classification behavior of feedforward networks an analysis is given of the empiricist vs. rationalist debate on the possibility or impossibility of induction. The analysis aims to show that forms of induction are possible but not without certain given forms of structure and representation. The analysis is then extended to cover the role of representation in pattern recognition, and the nature of the underlying forms of processing, aiming to show that models of cognition derive from a specific relation between dynamics and computation. These examples illustrate that an interaction between modeling and the discussion on foundational issues could be beneficial for both and could open new avenues in the philosophical and foundational debate not available in previous times.

1. Introduction

Foundational issues related to pattern recognition and learning have been topics of debate since the beginning of philosophy. But with the development of the computer, they have also turned into topics of scientific research.

The transition from philosophy to science is not unique for pattern recognition and learning. Similar transitions have occurred in astronomy, physics and other sciences. Yet, compared to these sciences, foundational issues related to pattern recognition still remain to be debated in a philosophical way as well. This could be because they concern the nature of human cognition. An example is the debate between rationalism, and the related notion of nativism, and empiricism. The debate can be traced back to antiquity (Plato) but is still a controversial issue today (e.g., [19]).

Here I aim to argue and illustrate that the scientific approach to pattern recognition can provide new information to influence the philosophical debate on foundational issues. In turn, this could inform the scientific approach to pattern recognition as well.

As an example, consider the role of induction in epistemology and philosophy of science. Induction can be seen as a key notion in the history of empiricism, ranging from Bacon in the 16th century to (logical) positivism, behaviorism and (part of) connectionism in the 20th and 21st century. The basic tenet of empiricism, as exemplified in epistemology and philosophy of science, is that we acquire knowledge exclusively through experience. Using induction, we can generalize our experience to arrive at law-like regularizations and categorizations. In turn, these can be used to explain our experiences (e.g., as belonging to a given category) or to predict new experiences. However, although induction provides categorizations and law-like regularities, these are entirely based on experience according to empiricism. No innate forms of knowledge would be needed.

According to rationalism this view on induction is untenable. An argument against it is given by the Goodman paradox (e.g., [4]). A simple example of this paradox was given by Hempel [4]. Consider a set of observations relating two variables. Assume that they can be plotted as a set of points in a two dimensional space, with one variable on the *x*-axis and the other on the *y*-axis, and that the points can be linked by a linear line. The question is whether this linear line is an induction provided solely on the basis of the set of points itself. The argument of the Goodman paradox is that this cannot be the case,

☆ This paper has been recommended for acceptance by Marcello Pelillo.

* Corresponding author at: Technical Cognition, CPE, CTIT, University of Twente, P.O. Box 217, Enschede, 7500 AE, The Netherlands. Tel.: +33534894637, +31 (0)53 489 4637; fax: +31 (0)53 489 4241.
E-mail address: f.vandervelde@utwente.nl

http://dx.doi.org/10.1016/j.patrec.2015.02.008
0167-8655/

because the linear line linking the points is just one of the choices that could be made. Another choice, for example, would be a sinusoidal curve linking the points in an oscillatory manner. The choice between these two (and other) options cannot be based on the data alone. Instead, it derives from a prejudice that precedes the induction (for example, the prejudice to choose the more 'simple' curve).

In other words, induction is impossible without some set of prejudices that precede it. This view underlies the reintroduction of rationalism (and with it nativism) in cognitive science in the 20th century, as exemplified by the notion that the structure of human language is innate (e.g., [3]). The nativism in modern cognitive science opposed the empiricism of behaviorism. In turn, however, the nativism in cognitive science resulted in an empiricist reaction based on connectionism. For example, in their book *Rethinking Innateness* Elman et al. [6] aim to show that cognitive processing as found in language can emerge from general purpose learning mechanisms instead of domain specific innate cognitive structures.

In his critique of *Rethinking Innateness* (RI), Fodor [7] argued that RI does not oppose all forms of innateness. That is, the notion that cognition results from an interaction between endogenous (innate) and exogenous information is not in dispute. What is in dispute is the nature of the endogenous information. RI would accept innateness of a learning mechanism but not the innateness of representational content. Given the need for prejudices in induction as outlined above, these prejudices could then result from an innate leaning mechanism, but would not consist of forms of innate representational content.

As argued above, the scientific investigation of pattern recognition could provide new tools to investigate foundational issues as well. One such tool could be the (mathematical or computational) analysis of learning and processing in connectionist (or neural) networks. Networks have been extensively used for pattern recognition. Specific examples are feedforward networks. Fortunately, a mathematical analysis of the classification behavior of these networks exists. This analysis could shed more light on the need for prejudices in induction and the nature and role of these prejudices.

Furthermore, the analysis of the pattern classification behavior of networks could also shed a light on foundational issues related to the underlying forms of processing and representation, which have also been debated in the course of history and in recent times.

In the sections below, network analysis is used to discuss foundational issues related to learning, processing and representation, beginning with an analysis of classification and induction with feedforward networks.

2. Feedforward networks as universal approximators

In mathematical terms a feedforward network implements a function between an input vector space (the 'domain' of the function) and an output vector space (the 'range' of the function). Hornik et al. [11] published an important theorem about feedforward networks. They showed that feedforward networks are 'universal approximators' for a wide class of functions. This means that for any function in this class one can find a feedforward network that approximates that function within any desired level of accuracy.

The fact that feedforward networks are universal approximators has sometimes been interpreted to mean that feedforward networks are equivalent to Turing machines. If so, one could calculate a function with a Turing machine and then approximate it with a feedforward network. However, the assumed equivalence between feedforward networks and Turing machines is not correct, because it ignores an important qualification in the theorem of Hornik et al. [11]. In particular, the universal approximator ability of feedforward networks is limited by the theorem to so-called 'compacta', that is, to closed and bounded subsets (intervals) of the input domain of the function. This limitation provides information about the induction abilities of feedforward networks.

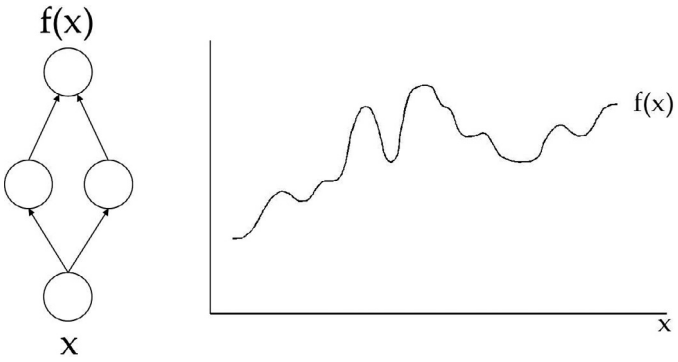

Fig. 1. A feedforward network as a function approximator.

The ability of feedforward networks to approximate functions, and the limitations of this approximation, can be illustrated with a feedforward network that approximates a continuous function $f(x)$ of a single variable x. An example of such a function is presented in Fig. 1, together with a feedforward network that could be used in the approximation. The network succeeds to approximate the function $f(x)$ if the network produces a value in its output neuron that is close to the value of $f(x)$ when the variable x is presented to its input neuron. The mathematical background of the theorem of Hornik et al. [11] shows how this can work.

The proof by Hornik et al. that feedforward networks are universal approximators is based on the Stone–Weierstrass theorem, which is itself a generalization of a theorem originally discovered by Weierstrass in the 19th century (e.g., see [23]). The Weierstrass theorem states that a continuous function $f(x)$ can be approximated ('uniformly') on an interval $[a, b]$ of its input domain by a series of polynomials. Thus, there is a (indefinite) series of the form $c_0x^0 + c_1x^1 + c_2x^2 + c_3x^3 + \ldots$ that approximates $f(x)$ for all values of x within $[a, b]$. The more terms are added to the series, the better the approximation will be. The limitation of the approximation to a finite and closed interval $[a, b]$ is the basis for the fact that feedforward networks can approximate functions only on limited subsets of the domain.

The reason for the limitation of an approximation is found in the coefficients c_0, c_1, c_2, c_3, etc., of the polynomial terms. The approximation of a given function requires a specific set of coefficients, unique for that function. To obtain these coefficients, or connection weights in the case of a feedforward network, information is needed about the approximated function. For instance, in the case of the backpropagation learning algorithm, function values are used as the desired output of the network, and the connection weights of the network are modified until the actual output of the network matches the desired output. Only a finite set of desired output values can be used in this way. The upper and lower limits of this set determine the interval on which the network can approximate the function.

Furthermore, the Weierstrass theorem demands that the function is known sufficiently ('globally') over the entire interval on which the function is approximated [2]. Hence, using scattered information about the approximated function on a given interval is in general not enough. Fig. 2 (left) illustrates this with the case in which the network in Fig. 1 attempts to approximate the function $f(x)$ on a large interval, after learning only a few function values (at x_1, x_2, x_3, x_4 and x_5) to determine the connection weights. With these function values, there are wide gaps within the interval on which the network has no information it could use to approximate $f(x)$. Fig. 2 (right) illustrates that the network could just as well use the learned function values to approximate a different function like $g(x)$, because this function has the same function values at x_1, x_2, x_3, x_4 and x_5. Because numerous functions like $g(x)$ exist, the chance that the network will approximate $f(x)$ in the gaps between x_1, x_2, x_3, x_4 and x_5 is zero.

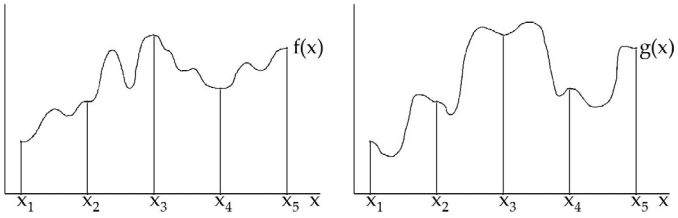

Fig. 2. Different functions can be learned with the same sparse set of the variable x (x_1, x_2, x_3, x_4 and x_5).

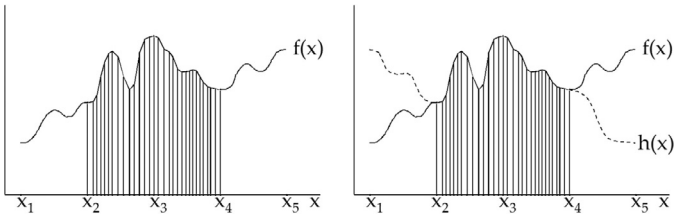

Fig. 3. Left: The feedforward network can approximate $f(x)$ in the interval x_2 to x_4 based on a sufficient set of learning values in this interval. Right: A function $h(x)$ that equals $f(x)$ for x_2 to x_4, but differs from $f(x)$ outside this interval.

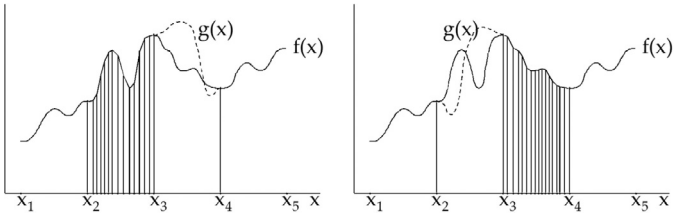

Fig. 4. Catastrophic interference with feedforward networks.

Thus, for the approximation to succeed, a sufficient set of function values within the interval has to be known, so that the remaining values can be approximated by interpolation between the known values. Fig. 3 (left) illustrates this for the function $f(x)$. When the network learns to approximate $f(x)$ with a sufficient set of function values in the interval x_2 to x_4, it can approximate $f(x)$ for the remaining values within this interval by means of linear interpolation. It is clear that the approximation will be more accurate when more function values within this interval are used for learning. In this manner, the network is a universal approximator.

However, it is also clear that the approximation of $f(x)$ is limited to the interval x_2 to x_4. Fig. 3 (right) illustrates this fact with the (dashed) function $h(x)$. This function is equal to $f(x)$ within the interval x_2 to x_4, but differs from $f(x)$ outside this interval. Again, there are numerous functions that are equal to $f(x)$ within interval x_2 to x_4, but which are different from $f(x)$ outside that interval. The network cannot distinguish between $f(x)$ and these functions when only function values within the interval x_2 to x_4 are used to learn the function $f(x)$, so it will fail to approximate $f(x)$ outside this interval.

The restriction of the Weierstrass theorem (a function can be approximated only on a given interval) also limits the learning behavior of feedforward networks. For instance, a phenomenon known as catastrophic interference occurs when a feedforward network tries to learn new information. The attempt to do so eliminates the ability of the network to produce the previously learned information (e.g., [16]).

Fig. 4 illustrates the occurrence of catastrophic interference for the network in Fig. 1. Suppose the network first learns to approximate $f(x)$ within the interval x_2 to x_3 before it learns to approximate this function in the rest of the interval (x_3 to x_4). This kind of learning is quite natural in cognition. We learn incrementally, in a step by step manner. As a result, the network can approximate $f(x)$ within the interval x_2 to x_3, but not yet in the interval x_3 to x_4. In that interval, the network could just as well approximate the function $g(x)$ or any other function different from $f(x)$.

Then, the network learns to approximate $f(x)$ in the interval x_3 to x_4. However, as a consequence of learning, the connection weights in the network are 'tuned' to produce the function values used for learning in the interval x_3 to x_4. At that moment, all previous 'tuning' of connection weights is lost. So, the network will no longer approximate $f(x)$ in the previous interval x_2 to x_3, because it could now just as well approximate $g(x)$ in this interval, or any other function different from $f(x)$. The only way for the network to approximate $f(x)$ in the interval x_2 to x_4 is to learn this interval in its entirety (as in Fig. 3), or to relearn $f(x)$ in the interval x_2 to x_3 when it learns to approximate $f(x)$ in the interval x_3 to x_4.

2.1. Feedforward networks and induction

The way feedforward networks can learn to approximate a function (or categorize their inputs) illustrates the interaction between endogenous (innate) and exogenous information. To approximate the function, function values need to be given to the network. Furthermore, they have to be sufficient number of them for the approximation to succeed. Other function values can be induced by extrapolation between the learned values. But successful extrapolation beyond the function values given is difficult due to the number of alternatives available.

The prejudices that guide the learning behavior clearly consist of the learning mechanisms that allow the network to learn the given function values by adjusting its connection weights. These learning mechanisms and the structure of the network need to be given (although a part of the network structure might perhaps arise in the learning process itself). So there is an innateness of a learning mechanism as accepted by RI.

However, there are also forms of innateness of representational content. This is clear from the mathematical requirements for the approximation by the network, such as the compactness and density of the function values on which the approximation can succeed. It also consists of the use of squashing functions with which each node needs to transform its input into an output [11]. These squashing functions transform their (potentially unbounded) input space to a limited output space. In this way, they discard or reduce information, which is necessary for the approximation (or classification) to succeed.

But the network is also capable of learning new classifications, based on the input given, and to successfully induce new function values when the input information is sufficiently large. This behavior contrasts the most radical version of rationalism, as perhaps found with (e.g., [7]), which seems to argue that new categorizations (e.g., new concepts) cannot be learned but have to be innate.

2.2. Hierarchical networks and classification

The feedforward or hierarchical networks illustrated in Figs. 1–4 illustrate a simple case of pattern recognition. But they can be extended to more complex examples. First, it is straightforward to extend them to multidimensional forms of pattern classification. In that case, more input and output nodes are needed. But the learning behavior extends to these added dimensions in a straightforward manner (the theorem of [11], is in fact stated for multidimensional networks).

Another extension is found in relating feedforward networks to pattern classification in the human (and primate) cortex. Recognition of familiar patterns occurs fast, suggesting a feedforward process, based on a large set of hierarchical layers, running from the retina to neurons in the inferotemporal cortex and beyond (e.g., [5]).

Serre et al. [25] implemented aspects of pattern recognition in a model based on the visual cortex. The model shares many of the features as found with the network illustrated in Figs. 1–4. Added

features are the use of Gabor filters to provide input representations (line orientations) and the H-max operation, which eliminates all input activation except the largest one.

In machine learning, hierarchical models such as deep learning networks (e.g., [10]) are among the most sophisticated pattern learning methods to date. Here restricted Boltzmann machines (RBMs) are used to pre-train the network, i.e., to give it initial connection weights so that the network does not get stuck in local minima when error backpropagation is applied. In the RBMs the representations that the input and output layers can have are given and the connection weights and the distribution of representations in the output layer are then changed so that the energy of the learned pattern is reduced compared to the energy of undesired patterns.

In the examples of hierarchical networks given here the learning methods are given and, so it seems, also forms of representational content. This raises the question of whether it is possible to make a distinction between these two issues. That is, is it possible to have a learning mechanism that does not require certain forms of representation? Consider again a squashing function. In the case of a feedforward classification it is needed to somehow reduce the amount of information (i.e., classification results from information reduction). But the way in which that is done also determines the way you store information, and thus it would seem the representational content the model can have. For example, there are squashing functions that simply reduce the incoming information (activation) into two classes, e.g. 0 and 1. This clearly affects the representations that a system could have. The representations as used in the RBMs discussed above also provide a direct link between a learning mechanism and given representational content. One cannot function without the other.

3. Representation and computation

The behavior of the network in Fig. 1 illustrates other foundational issues about cognitive science, including pattern recognition, that have been debated intensively. It seems that the network learns to *represent* the function it approximates (within the compact subset in which it learns the function). And it seems that the network implements a computational program (algorithm) to do so. The latter assumption is strengthened by the observation that training networks to classify functions is executed by a computer program and simulated on a computer.

The idea that cognition depends on forms of computation is a key notion within cognitive science (e.g., [17]) and one of the corner stones of the rationalistic view in cognitive science (e.g., [9]). However, this view is not without dispute. Searle [24], for example, argued that all mental processes are brain processes, which are nonrepresentational and non-computable by nature. The question arises what these nonrepresentational and non-computable processes would be and how they can be used for pattern recognition. Given its importance for cognition (both human and artificial), it would be important to see how pattern recognition could be achieved without representation and computation. Or, alternatively, what would be missing in representation and computation to achieve pattern recognition. Concerning the latter, even Fodor [8] argues that computation is incomplete to account for human cognition and pattern recognition in particular.

An alternative to representation and computation could be found in dynamical systems. The views on the relation between dynamics and cognition vary between two extremes. For classical cognitive science or psychology, cognition is based on representation and computation, which are not affected by dynamics. At best, dynamics is related to the way in which cognitive processes are implemented, but this is unimportant for understanding cognition (e.g., [9]).

In contrast, in the dynamical approach to cognition (e.g., [21]) computation and representation are not important, if not misleading, for understanding cognition. Instead, in this view, every cognitive process is based on a dynamical process in which classical computation and representation do not play a role. This could also be the basis for the mental processes as intended by Searle, because they would depend on brain processes, which might operate as dynamical systems.

To investigate issue, I will discuss the relation between computation and dynamics is discussed from a network perspective.

3.1. Computation and dynamics

The study of dynamics started with the mathematical description of the motion of a planet around the sun. In this case, two bodies attract each other and the motion of both is influenced by this attraction. The focus on motion reflects the importance of time in dynamics. In general, dynamics describes how a system of interacting elements evolves in time (e.g., [13]). From this perspective, a neural network (and the brain) is a dynamical system as well. This raises the question of how the dynamics of neural networks affects cognition.

Classical cognitive science has a straightforward answer to this question. In their discussion of connectionism, Fodor and Pylyshyn [9] argued that, at best, neural networks could provide a description of the way in which cognitive processes are implemented in the brain. But, in their view [9, p. 65]:

> … the implementation, and all properties associated with the particular realization of the algorithm that the theorist happens to use in a particular case, is irrelevant to the psychological theory; only the algorithm and the representations on which it operates are intended as a psychological hypothesis.

The independence of cognition from implementation is a key notion in classical cognitive psychology (e.g., see [14,20]). Indeed, some neural modelers aim to show explicitly that they do not just provide an implementation theory of classical cognition (symbol manipulation) because they do not use explicit conceptual representations in compositional structures (such as symbols in classical compositional structures, e.g., [26]).

It is important to note that Fodor and Pylyshyn do not deny that there is also an issue of implementation. They assert only that this level has no consequences for the level of cognition, that is, the regularities of the latter are not determined by the first. This is related to the distinction between the computational, algorithmic and implementational level as proposed by Marr [15]. This distinction has frequently been cited to argue that psychology should not be concerned with implementation (e.g., [12]). It has also been used, for example, to argue against connectionism as being merely an implementation theory (e.g., [1]). However, Marr himself in fact made a distinction between type I and type II theories. For type I theories the distinction between levels as he described holds, but it does not hold for type II theories (see [30]).

So, the assumption of classical cognition is that theories of cognition are type I theories in Marr's sense. Because cognition depends on computations (algorithms) and representations, the kind of machine on which an algorithm runs is irrelevant. Any machine with sufficient computing power will do, including machines as slow as the Turing machine. Its slowness has no effect on cognition according Fodor and Pylyshyn [9, p. 55], because:

> … the absolute speed of a process is a property par excellence of its implementation.

But in many real-life situations the process of selecting the location of an identified object has to occur within a given time limit. Traffic comes to mind as an example in modern times, but the interactions with prey and predators in ancient times provide equally good examples. In each of these cases, there is a dynamics of cognition that is dictated by the dynamics of the outside world. A prey will be long gone before a cognitive agent equipped with a genuine Turing machine has identified its location, so the agent would die of starvation.

Or, it would have been eaten long before it could have identified the location of a predator.

These examples indicate that computations (algorithms) and representations are incomplete as a basis for cognition. Here, it is important to realize that there is no precise mathematical definition of the notion 'algorithm'. This results from the fact that there is no mathematically precise definition of the notion of an 'effective procedure'. But there is a precise definition of 'recursive' (or computable) functions (e.g., see [22]). So, if you want to know if two algorithms or effective procedures (e.g., the Turing machine vs. Lambda calculus) are equivalent, you compare the (set of) recursive functions they compute.

Thus, if Fodor and Pylyshyn argue that the architecture of cognition is comparable to that of a Turing machine of von Neumann architecture, they are in fact referring to the set of functions these architectures can compute. This is the only sensible comparison that can be made between such distinct architectures. To say that implementation is not relevant is in fact saying that the function computed (e.g., the function that distinguishes all correct sentences of a natural language from the incorrect ones) is not affected by the level of implementation. The fact that you can have different algorithms for the same function in fact corroborates this view: if algorithms can be different but have the same effect, then the implementation of one of them will not be relevant either.

However, for cognition the "machine" that implements an algorithm is relevant, because the absolute speed with which an algorithm is executed has to match the dynamics dictated by the environment. In turn, this requirement could select the kind of processes that can be executed on this machine. After all, although an algorithm can be implemented on different machines, the converse is not necessarily true. Not every machine can implemented any given algorithm. Instead, the machine determines the nature of the algorithms that it can implement. So, the requirement that the cortex has to execute cognitive processes fast enough to meet the dynamics of the outside world could influence the kind of processes that it can execute.

In this way, evolution shaped the nature of cognition. First and for all, evolution selected the execution of an algorithm. It selected those "machines" that could identify preys and predators fast enough, because a cognitive agent that identified the location of a prey or predator too slow could not survive. In turn, these "machines" selected the kind of algorithms they could produce. The feedforward network in Fig. 1 provides an example. Feedforward networks are among the fastest operating networks, which can also learn to classify objects.

3.2. Functions and flow

The relation between computation and dynamics, and its effect on cognition, can be described in more formal (mathematical) terms. Computation theory is a branch of mathematics that, informally, studies which mathematical functions can be generated by a "procedure" (or algorithm), and how this can be done (e.g., [22]). The details do not matter here, but it is highly relevant to note that computation theory deals with mathematical functions. This is in fact the basis of Fodor and Pylyshyn's assertion that implementation is irrelevant for cognition (see above).

In general, a mathematical function is a collection or list of pairs. The first member of each pair is the input of the function, and the second member is its output. For some functions, it is possible to give a rule that describes the list of all function pairs, such as the rule $f(x) = x + 1$. A function is computable if a procedure can be given, such as the Turing machine, that produces the correct output of the function when an input is given. Computation theory has shown that the set of computable functions is a genuine subset of the set of all mathematical functions.

So, a procedure or algorithm computes a function if it gives the correct output for a given input. But which output is correct is determined by the function description (i.e., its list of input–output pairs or, occasionally, its function rule). This immediately shows why implementation factors such as speed of execution are irrelevant: they are not included in the function description. The algorithm that computes $f(x) = x + 1$ has to produce 5 for the input 4, regardless of how long it takes. If it produces 6 or 3 for that input, it fails to compute the function.

Therefore, Fodor and Pylyshyn's [9] claim that implementation is irrelevant for cognition is equivalent to the claim that every cognitive problem that a cognitive agent has to solve can be described entirely in terms of computable functions. Indeed, this equivalence is implicit in the phrase "only the algorithm and the representations on which it operates are intended as a psychological hypothesis", quoted above.

However, a strong argument can be made for the claim that not every cognitive problem can be described entirely in terms of functions, because a function is nothing more than a static list of input–output pairs. Consider the classification problem of deciding whether an animal is a prey or a predator. This problem clearly has a functional aspect to it. The input is the encountered animal, and the output is the classification "prey" or "predator". But our ancestors would not have survived their harsh environment if they had handled this problem only as a function. The dynamics of the outside world, which is beyond our control, makes the difference. The decision "prey" or "predator" has to be both rapid and safe, and being safe could entail that you could even make a decision that is functionally incorrect (but not random). Cortical dynamical systems that interact in real time with the environment have to, and can, make such decisions.

Furthermore, the interaction with the environment is an ongoing process. Prey or a predator could be moving while the classification process is executed. Dynamical systems provide an account for cognitive processes in a time depended manner, needed to understand the interaction with the environment.

Yet, the functional aspect of a cognitive process, and its link with computable functions, is not lost. Fig. 5 illustrates the relation between dynamical systems and mathematical functions. The evolution in time of a dynamical system can be depicted in a state space (or phase space). In general, a dynamical system evolves along a trajectory in its state space, beginning at some initial state. The trajectory in Fig. 5a begins in state x_0 and moves to states x_1, x_2, x_3 and x_t. The subscript t in x_t refers to the time it takes to go from x_0 to x_t along the trajectory.

Fig. 5b shows that a function $g(x)$ can be defined that maps the state x_0 to the state $g(x_0)$ in state space. Other functions like $h(x)$, $p(x)$ and $q(x)$ map x_0 to the states $h(x_0)$, $p(x_0)$ and $q(x_0)$ respectively.

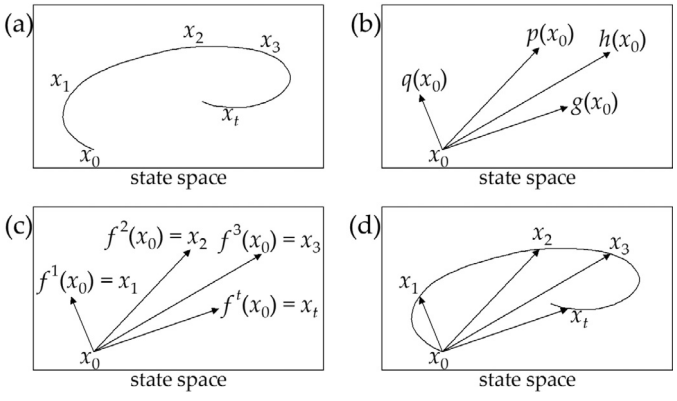

Fig. 5. Dynamical systems and functions. In (a), a dynamical system follows a trajectory in state space. In (b), functions that map between states in state space. In (c), equivalence between functions and trajectory states. In (d), a flow of a dynamical system (group of functions, with parameter t).

Fig. 5c shows that the functions $f^1(x)$, $f^2(x)$, $f^3(x)$ and $f^t(x)$ can be defined such that $f^1(x_0) = x_1$, $f^2(x_0) = x_2$, $f^3(x_0) = x_3$, and $f^t(x_0) = x_t$, that is, they are functions that map the initial state x_0 to the states x_1, x_2, x_3 and x_t along the trajectory, as illustrated in Fig. 5d. The superscript labels refer to the time parameter t that governs the evolution of the dynamical system. In general, $f^t(x_i) = x_i + t$, which means that x_i is mapped to $x_i + t$ in time t for any x_i and $x_i + t$ along the trajectory.

The time parameter t entails that there is a strong relation between the functions $f^1(x)$, $f^2(x)$, $f^3(x)$ and $f^t(x)$. For example, $f^3(x_0)$ maps x_0 to x_3 in 3 s. But we could also start in x_2 and apply $f^1(x_2)$, which maps x_2 to x_3 in 1 s. Because $f^2(x_0) = x_2$, we get:

$$f^3(x_0) = f^1(f^2(x_0)) \quad \text{or } f^{1+2}(x_0) = f^1(f^2(x_0))$$

The equivalence $f^{1+2}(x_0) = f^1(f^2(x_0))$, or $f^{t+s}(x) = f^t(f^s(x))$ in general, shows that the effect of the functions is additive. The function f^{t+s} achieves the same result as the functions f^s and f^t applied in sequence. This property means that the functions $f^1(x)$, $f^2(x)$, $f^3(x)$ and $f^t(x)$ form a mathematical group with parameter t, known as the *flow* of the dynamical system (e.g., [13]).

The flow of a dynamical system can be used to describe the relation between dynamics and computation. Suppose a cognitive agent uses the dynamical system in Fig. 5 to classify an animal as prey or predator. The presence of the animal prepares the dynamical system in the state x_0 and its classification as prey or predator occurs at time t, when the dynamical system is in the state x_t. The classification of the animal as prey or predator can be seen as the solution of a functional problem, with state x_0 as the input and state x_t as the output. The dynamical system has solved this classification by executing the function $f^t(x_0) = x_t$.

The classification of the animal as prey or predator could also be achieved by the function $g(x)$ in Fig. 5b, assuming that $g(x_0) = x_t$. This classification function $g(x)$, and the other functions $h(x)$, $p(x)$ and $q(x)$ in Fig. 5b, could be computable functions. If so, these functions are computed in a computer simulation of the dynamical system, so $g(x_0) = x_t$ is the computer simulation of $f^t(x_0) = x_t$.

According to Fodor and Pylyshyn [9], the algorithm and representations that produce $g(x)$ account for the cognitive ability to classify an animal as prey or predator, and $f^t(x)$ is just an implementation of $g(x)$. But this ignores the importance of the dynamic interaction with the environment reflected in the parameter t of $f^t(x)$. Not every implementation of $g(x)$ succeeds in producing $g(x_0) = x_t$ in time t. An implementation that fails to do so, cannot sustain a cognitive agent in its interaction with the environment. Therefore, to account for the cognitive ability to classify an animal as prey or predator in the interaction with the environment, the complete description of $f^t(x)$ is needed, and not just its resemblance to $g(x)$.

3.3. Dynamics and computation

Perhaps any relation between $f^t(x)$ and $g(x)$ is irrelevant for understanding cognition. This, in short, is the view of dynamical approach to cognition (e.g., [21]), and perhaps that of Searle [24]. The problem with this view is that there are numerous dynamical systems that have no cognitive abilities whatsoever. The solar system, for example, or the coffee in a cup of coffee (or even the cup itself) are dynamical systems without cognitive abilities. How can we distinguish between these dynamical systems and the ones that do have cognitive abilities, such as pattern recognition? It seems that the only way to do this is to consider the resemblance between $f^t(x)$ and $g(x)$. For example, the fact that $g(x)$ is (or has to be) a classification function provides constraints for the dynamical system that produces $f^t(x)$.

Fig. 6 illustrates this point with the "chaotic attractor" of a three dimensional (deterministic) dynamical system (modeled after the well-known Lorenz attractor, e.g., [27]). A region in the state space of a dynamical system is an attractor if trajectories in the state space move towards that region. The fact that a dynamical system has

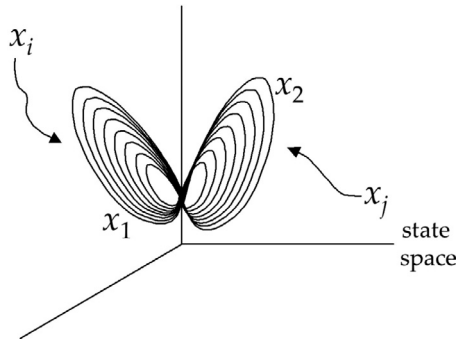

Fig. 6. Illustration of a "chaotic attractor" in the three-dimensional state space of a dynamical system. All initial states (e.g., x_i, x_j) of the dynamical system result in trajectories that evolve towards the attractor. But within the attractor (e.g., x_1, x_2), the dynamical system is unpredictable.

one or more attractors means that part of its behavior is predictable. No matter where it starts in its state space, it will evolve towards an attractor.

In this case, all initial states, such as x_i and x_j, produce trajectories that move towards the attractor illustrated in Fig. 6. But all predictability is lost within the attractor, even though the dynamical system is deterministic. The loss of predictability is due to an extreme sensitivity to even the smallest variations. Suppose, for example, that a trajectory exist within the attractor from x_1 to x_2. Because the system is deterministic, it will produce the same trajectory whenever it is in the state x_1. But even a very small deviation from this state can result in a trajectory that does not move towards x_2, or any state close to it. The same is true for trajectories between initial states and states within the attractor. Suppose there is a trajectory from x_i to x_1. Again, even a very small deviation from x_i can result in a trajectory that does not go to x_1 (although it does move into the attractor).

Now suppose a cognitive agent wants to use this dynamical system to classify an animal as a prey or predator. We could assume that when the animal is a prey, the dynamical system begins in x_i and moves toward x_1 within the attractor, and when the animal is a predator, it begins in x_j and moves toward x_2 within the attractor. The states x_1 and x_2 are thus the classification states for prey and predator, respectively, and they could be used to initiate a response (e.g., catching the prey or running away from the predator).

However, the classification behavior of this dynamical system is totally unreliable. The next time it encounters an animal, it can classify it as a prey, for example, only if it starts exactly in the state x_i. But the chance that a dynamical system can start again in exactly the same (real-valued) state is zero. Small deviations from x_i will inevitably occur (e.g., due to Brownian motion). At best, it will start in a state somewhere near x_i, which, in this case, can easily result in a trajectory that does not move to or near x_1.

The inevitability of small variations shows that cognitive processes like classification cannot depend on mappings between single states in the state space of a dynamical system. It can succeed only if a region around an initial state like x_i is mapped to a region around a classification state like x_1. Only in this way is there a sufficient chance that the behavior of the dynamical system is replicable. Without that, a cognitive system cannot produce purposive behavior or learn from its experiences.

Again, a cognitive agent using the chaotic attractor for classification serves to illustrate this point. Assume that the first time the agent encounters a predator, the state space changes from x_j to x_2, and the latter state is used as a trigger to run away. The next time the agent encounters the same predator, though, it will not start at x_j but somewhere near that state. For these cases, there is no guarantee that it will end up at x_2 in its state space or close to it. But without a reliable relation between an initial state and an end state, a cognitive agent

cannot rely on an end state as a trigger for an action. Furthermore, it cannot learn either, and the experience of any encounter with the predator is lost.

If a cognitive agent uses the entire chaotic attractor as a classification state, the dynamical system produces reliable behavior. But then the agent does not make meaningful distinctions in its state space, because all initial states move towards the same attractor.

A more interesting case occurs when the cognitive agent uses each of the two "wings" of the attractor as different classification states. It cannot use them for a reliable classification of prey or predator, because the dynamical system moves unpredictably from one wing to the other (so the behavior of the agent would change unpredictably from catching or fleeing if it used the wings as classification states). But the cognitive agent could use each wing to make a decision. For example, "yes" if the trajectory of the dynamical system is in the left wing at a given moment, and "no" if it is in the right wing. The dynamical system will be in one of the two wings at a given moment (after its initial behavior), but it is unpredictable in which of the two. Thus, the cognitive agent would use the wings of the attractor as a random generator, to decide between "yes" and "no" in a probabilistic manner.

3.4. Cognitive dynamics

The example of the chaotic attractor shows that the nature of cognition imposes constraints on the nature of dynamical systems that can be used in cognitive processes. To be reliable, the behavior of a dynamical system cannot depend on single states but has to depend on regions in state space. To produce meaningful behavior, the dynamical system cannot map all initial states to the same end state, but instead has to make distinctions between regions in its state space. These constraints entail that the dynamical system cannot be a conservative (closed) system, but has to be open and dissipative.

In a conservative dynamical system, the volume of the state space does not change during its evolution. Examples are the energy-conserving systems of classical mechanics, known as Hamiltonian systems (e.g., [13]). Penrose [18] used the energy-conserving property of these systems as an argument for the claim that cognition depends on quantum physics, instead of computation based on classical physics.

An important characteristic of energy-conserving systems is described by the theorem of Liouville, which states that the volume of any region in state space remains constant during the evolution of the system. This characteristic can result in periodic behavior, but more generally it results in the behavior illustrated in Fig. 7a. Here, the state space evolution of an energy-conserving system starts with the initial state given by a region around x_0. Due to Liouville's theorem, the volume of this region remains constant, but it spreads out over the state space. Eventually, the trajectories emerging from the initial region will return close to the states were they began (e.g., [27]).

The volume preserving property of energy-conserving systems makes them unsuited for cognitive processes. Either they exhibit repetitive behavior, or they cannot make reliable distinctions in their state space. For example, assume that a cognitive agent uses an energy-conserving system for classifying an animal as prey or predator. When the animal is a prey, the dynamical system starts in one initial region in state space, and when the animal is a predator, it starts in another. The volumes of both initial regions remain conserved during the evolution of the system, but they become intermingled in a way that prevents a reliable distinction between the classification states for prey and predator.

In a dissipative dynamical system, as illustrated in Fig. 7b, the volume of an initial region in state space decreases as the system evolves in time. In general, most trajectories in a dissipative system evolve toward an attractor (e.g., [27]), and the volume of the attractor is much smaller than the volume of its initial states (i.e., the basin of the attractor). When the system has two or more attractors, it can make meaningful distinctions in its state space, in which different internal regions evolve toward different reliable end states (e.g., classification states in a classification task).

Dissipative dynamical systems with more than one attractor are highly suited for cognitive processes. For example, a cognitive agent can use a dissipative dynamical system to classify an animal as prey or predator, because it can divide the state space into two attractor basins, to make a reliable distinction between the classification states for prey and predator.

3.5. Dynamics and networks

A dynamical system can dissipate energy when it is in open contact with an environment. It uses the environment to "dump" the decreasing volume of its state space as it evolves in time (like heat that radiates to open space in a cloudless night). In contrast, a conservative dynamical system is a closed system that does not exchange energy (state space activation) with an environment.

Fig. 8a illustrates an (almost) closed dynamical system consisting of interacting (white hexagonal) cells. Most cells are enclosed by other cells, except the cells at the outside, which are also in contact with the outside environment. The center cell x can use only its surrounding cells to dump any of its state space volume, but these cells in turn also use cell x to dump their state space volume. Only the cells at the outside can use the outside environment to dump some of their state space volume. Eventually, some of the state space volume of the enclosed cells (such as x) trickles down to these outside cells, and

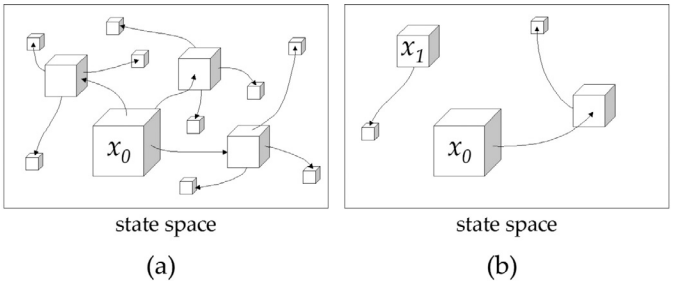

Fig. 7. Dynamical systems and evolution in state space: In (a), the evolution of an energy-conserving (closed) dynamical system. A region around an initial state x_0 in the state space does not decrease in volume, but spreads over the state space when time increases. In (b), the evolution of a dissipative (open) dynamical system. A region around an initial state x_0 in the state space decreases in volume, and evolves towards an attractor state when time increases. The system can make meaningful distinctions in its state space when other initial regions like x_1 evolve to other attractors.

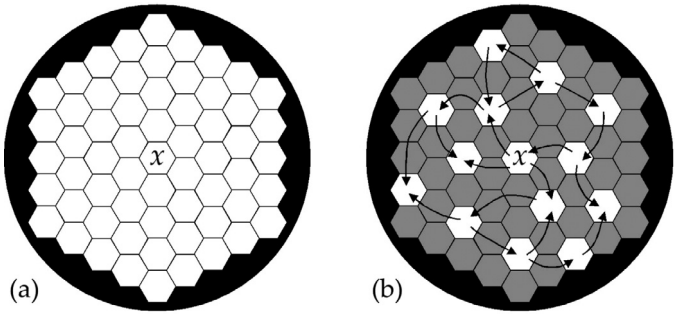

Fig. 8. Closed and open dynamical systems (of white hexagonal cells). In (a), most cells are enclosed by other cells. Only the cells at the border of the system are in contact with the outside environment (black circle). As a result, the dynamical system is basically closed. In (b), the white cells form an interconnected subsystem. Each enclosed white cell is now surrounded by grey cells (the internal environment). The dynamical system of white cells is now maximally open.

flows into the outside environment as well, but this will take a while to occur. So, in the short run, the state space of cell *x* and any other enclosed cell is practically conserved, as in Fig. 7a. In a large system of this kind, where the enclosed cells dominate, the entire system is practically a closed dynamical system.

In Fig. 8b the situation is very different. Here, the white cells form an interconnected subsystem in between the grey hexagonal cells. Each enclosed white cell is now surrounded by grey cells (the internal environment). So each enclosed white cell, such as the center cell *x*, can use the grey cells to dump their state space volume. As a result, each white cell is an open dynamical system, as in Fig. 7b. Of course, the system as a whole (white and grey cells combined) is of the same kind as the one in Fig. 8a. But now the white cells form an interconnected subgroup, which itself is maximally open. In this way, the functionality of this group is not affected by the closed nature of the system as a whole (provided the overall system can dump enough of its state space to the outside environment).

It is clear that the system of white cells in Fig. 8b form a network. It seems they would have to be a network, because each white cell has to be open and, at the same time, be in contact with other white cells to form a functional system. A white cell can achieve the first by its contact with the internal environment (i.e., its contact with the grey cells to dump its excess state space volume), and the latter by the use of its connections. Viewed in this way, a neural network is a maximally open dynamical system. In the brain, neurons are in contact with the internal environment of the brain (e.g., the fluids and support cells that surround them), which they can use to dissipate activation. At the same time, they are in contact with each other through their connections, which determines the functionality of the network.

The hierarchical networks discussed above illustrate the importance of dissipation as well. For example, consider the use of squashing functions in the classification network in Fig. 1. When a neuron uses a squashing function, it receives more activation than it passes on to other neurons. Therefore, the neuron has to dissipate the excess of activation. In the closed system of Fig. 8a, the white cell *x* can use only other white cells to dissipates activation, which in turn use cell *x* to dissipate their activation. But in the network structure of Fig. 8b, the white cell *x* receives activation from the other white cells only, and it can easily use its surrounding internal environment (the grey cells here), to dissipate activation. So, the white cells in Fig. 8b can operate with squashing functions, due to the network structure to which they belong.

Other examples of dissipation are the reduction of dimensions in deep learning networks (e.g., [10]) and the use of the H-max operation, which eliminates all input activation expect the largest one, in the model of the visual cortex of Serre et al. [25].

4. Overview and conclusion

The fact that a neural network can dissipate its activation and at the same time operate as a functional system relates to a number of issues concerning the foundation of pattern recognition and cognition in general.

For example, Penrose's [18] argument that the brain cannot be (in part) computational because of the Liouville problem is incorrect. Networks in the brain are not closed dynamical systems. In fact, they are maximally open due to the fact that each neuron is both connected to a network and can interact with its local (non-network) environment. Viewed in this way, the brain consists of two organs meshed into one: the brain as a network (white cells in Fig. 8b) and the environment supporting each neuron (the grey cells in Fig. 8b).

Also, Fodor and Pylyshyn's [9] argument that the level of implementation is irrelevant is highly disputable. If the brain has to resort to a network structure to combine functionality with activation dissipation then the network structure may be crucial for understanding cognition. Also, because cognition (and pattern recognition) do not just concern problems that can be solved entirely by calculating (implementing) classification functions. Instead, a classification function will be part of a dynamical system (flow) in which time is an important parameter. In turn, the time parameter imposes important constraints on the implementation (as acknowledged by Fodor and Pylyshyn). These implementational constraints could then impose constraints on the network architecture and thus its function. The feedforward network in Fig. 1 illustrates this point: feedforward networks are among the fastest operating networks, which makes them highly suitable for particular forms pattern recognition and learning, as illustrated in Figs. 3 and 4.

In a similar vein, the notion that cognition and thus pattern recognition is not representational (e.g., [21,24]) is highly disputable. The dissipative behavior of neurons, illustrated in Fig. 8b, seems to invite representational aspects of network behavior, in which the activation of a neuron is determined by network structure of which it is a part. An example is the use of squashing functions in Hornik et al.'s [11] approximation theorem.

When neurons operate in the manner of squashing functions they represent their input space in a reduced output space, thereby affecting the behavior of the overall network. As analyzed above, this affects the learning and inductive behavior of the network. Networks are capable of induction but they do need a given basic structure and forms of representation (e.g., as given by squashing functions). Furthermore, networks can implement computational functions as long as these functions are part of the flow that determines the behavior of the networks, as illustrated in Fig. 5.

The dissipative nature of networks could have consequences for the nature of neural network representation in language and high-level cognition as well (e.g., [28,31]), which in turn affects the learning abilities of these networks [29].

The fact that a number of issues about the foundation of pattern recognition and cognition in general can be discussed in terms of basic features of models such as feedforward networks and networks as dissipative dynamical systems justifies the hope that other foundational issues could also be discussed in this way (e.g., using more elaborate models). This interaction between modeling and the discussion on foundational issues could be beneficial for both and could open new avenues in the philosophical and foundational debate on pattern recognition and cognition in general not available in previous times.

Acknowledgments

I thank two anonymous reviewers for their valuable comments made on an earlier version of the manuscript. The work of the author was funded by the project ConCreTe. The project ConCreTe acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 611733.

References

[1] D.E. Broadbent, A question of levels: comment on McClelland and Rumelhart, J. Exp. Psychol. Gen. 114 (1985) 189–192.
[2] F.W. Byron, R.W. Fuller, Mathematics of Classical and Quantum Physics, Dover, New York, 1992.
[3] N. Chomsky, New Directions in the Study of Language and Mind, Cambridge University Press, Cambridge, 2000.
[4] N. Chomsky, J.A. Fodor, The inductivist fallacy: statement of the paradox, in: M. Piattelli-Palmarini (Ed.), Language and Learning, Routledge, London, 1980, pp. 259–261.
[5] J.J. DiCarlo, D.D. Cox, Untangling invariant object recognition, Trends Cogn. Sci. 11 (2007) 333–341.
[6] J. Elman, et al., Rethinking Innateness: A Connectionist Perspective on Development, MIT Press, Cambridge, MA, 1996.
[7] J.A. Fodor, In Critical Condition, MIT Press, Cambridge, MA, 1998.
[8] J.A. Fodor, The Mind Doesn't Work That Way, MIT Press, Cambridge, MA, 2000.
[9] J.A. Fodor, Z.W. Pylyshyn, Connectionism and cognitive architecture: a critical analysis, Cognition 28 (1988) 3–71.

[10] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507.
[11] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[12] M.S. Humphreys, J. Wiles, S. Dennis, Toward a theory of human memory: data structures and access processes, Behav. Brain. Sci. 17 (1994) 655–692.
[13] E.A. Jackson, Perspectives of Nonlinear Dynamics, vol. 1–2, Cambridge University Press, Cambridge, 1991.
[14] G.F. Marcus, The Algebraic Mind, MIT Press, Cambridge, MA, 2001.
[15] D. Marr, Vision: A Computational Investigation Into the Human Representation and Processing of Visual Information, Freeman, 1982.
[16] M. McCloskey, N.J. Cohen, Catastrophic interference in connectionist networks: the sequential learning problem, in: G.H. Bower (Ed.), The Psychology of Learning and Motivation, 24, Academic Press, New York, 1989, pp. 109–165.
[17] A. Newell, Unified Theories of Cognition, Harvard University Press, Cambridge, MA, 1990.
[18] R. Penrose, The Emperor's New Mind, Oxford University Press, Oxford, 1990.
[19] M. Piattelli-Palmarini, Language and Learning, Routledge, London, 1980.
[20] S. Pinker, How the Mind Works, Penguin, London, 1998.
[21] R.L. Port, T. van Gelder, Mind as Motion: Explorations in the Dynamics of Cognition (Overview), MIT Press, Cambridge, MA, 1995.
[22] H. Rogers, Theory of Recursive Functions and Effective Computability, MIT Press, Cambridge, MA, 1988.
[23] W. Rudin, Principles of Mathematical Analysis, McGraw-Hill, New York, 1976.
[24] J. Searle, The Rediscovery of the Mind, Harvard University Press, Cambridge, MA, 1992.
[25] T. Serre, A. Oliva, T. Poggio, A feedforward architecture accounts for rapid categorization, Proc. Natl Acad. Sci. 104 (2007) 6424–6429.
[26] T. Stewart, C. Eliasmith, Compositionality and biologically plausible models, in: M. Werning, W. Hinzen, E. Machery (Eds.), Oxford Handbook of Compositionality, Oxford University Press, Oxford, 2012.
[27] J.M.T. Thompson, H.B. Stewart, Nonlinear Dynamics and Chaos, Wiley, Chichester, 1986.
[28] F. van der Velde, M. de Kamps, Neural blackboard architectures of combinatorial structures in cognition, Behav. Brain Sci. 29 (2006) 37–70.
[29] F. van der Velde, M. de Kamps, Learning of control in a neural architecture of grounded language processing, Cogn. Syst. Res. 11 (2010) 93–107.
[30] F. Van der Velde, G. Wolters, A.H.C. van der Heijden, Marr contra Marr, Behav. Brain. Sci. 17 (1994) 681–682.
[31] F. van der Velde, Communication, concepts and grounding, Neural Netw. 62 (2015) 112–117.