MASTER OF SCIENCE IN PHYSICS OF COMPLEX SYSTEMS

POLITECNICO DI TORINO
SORBONNE UNIVERSITÉ
UNIVERSITÉ DE PARIS
UNIVERSITÉ PARIS-SACLAY

Biologically Plausible Learning Algorithm for Recurrent Neural Networks

Author:
Mattia Della Vecchia

Host Supervisor:
Vincent Hakim,
Theoretical neuroscience and biophysics team, LPENS

Internal Supervisor:
Andrea Pagnani,
DISAT, PoliTo

October 2021

Abstract

The very first works that gave light to the field of Artificial Intelligence were heavily influenced by the study of the brain, and carried out by collaborative efforts of computer- and neuro-scientists. Progressively, the models and methods proposed have distanced themselves from this approach, towards ones more oriented to informatics. In the last years, a resurgence of interest in works that bridge the two fields has taken action, thanks also to development of tools that allows to investigate the brain to an extraordinary level of detail. Biological mechanisms act as a natural source of inspiration for innovative techniques applicable to artificial networks. The project of this internship positions itself in this trend. The objective is to implement a training procedure for recurrent neural networks starting from considerations on the cerebellum. A form of Stochastic Gradient Descent has been proposed to be performed by this structure, involved in the coordination of motor actions. Activity of neurons is perturbed in a stochastic manner that results in a change of the global error. Synaptic plasticity is carried out integrating information about perturbations and error evolution. Two training algorithms for recurrent neural networks are taken as a base for investigation on the temporal dynamics and learning in this special class of networks, and are progressively modified to implement the mechanism proposed for cerebellar learning.

Contents

1 Introduction

The first efforts to have machines performing human-like tasks dates back to the beginning of the 1940s, with the development of the first computers. In 1943, it was firstly introduced the concept of neural networks (McCulloch and Pitts (1943)): systems for computations of logical functions vaguely inspired by the ones found in the brain.
The subsequent two decades were characterized by intense research in the emerging field of Artificial Intelligence (AI), with an eye open on biological networks. Researcher were later able to make these networks learn both in a supervised manner, with the introduction of the Perceptron (Rosenblatt (1958)), or unsupervised fashion, by incorporating and deriving underlying contextual statistics of the data presented (Hebb (1949)). Throughout the following decades, research in the field experienced oscillating periods of eclipse and resurgence, named winter and summer of AI.
The definite explosion of the subject and its implementations occurred during the early 2000s, thanks to strong advancements in computational power, an ever-growing production of data to be analyzed, and the proposal of a core idea in network architecture design: Deep Neural Networks. This evolved out of a rich line of research, named Parallel Distributed Processing (PDP) (Rumelhart et al. (1986)), whose efforts focused on the role of stochastic and highly parallelized information processing within biological neural networks based on a strong study of the human brain.
These new architectures are composed of a large number of layers between input and output, thus, the name Deep Networks. They are responsible for the top performances in almost every applications nowadays, and they have became a indispensable presence in many areas of research and technology.A Artificially intelligent systems have a boundless range of applications that allows to exploit the behemoth amount of data that we produce and have to analyze, with capabilities to perform useful and crucial tasks to a degree higher than human-level possibilities.

The revolution in the world of AI we have witnessed is driven by an approach heavily-oriented on computer science. Consequently, the principles of neuroscience, that played a crucial role in the foundation of the field, lost gradually their relevance. Multiple experts are calling for an urgency to reconcile the two fields, which have endured exponential developments in recent times, for further communications and collaborations. In the past years, many projects and studies implemented mechanisms observed in the human-brain in artificial neural networks to obtain state-of-the-art results (for a review Hassabis et al. (2017), notable examples are Gregor et al. (2015), Reed et al. (2015), Weston et al. (2015), Blundell

et al. (2016), Kirkpatrick et al. (2017)). I am in strong support of this line of research, and for this reason, I pursued an internship that would allow me to familiarize with elements and notions of both neuroscience and artificial intelligence.

1.1 Scope of the project

The interest for this project has its roots in the paper by Bouvier et al. (2018). It proposes a possible algorithmic implementation of learning in the cerebellum. This is a brain structure that is believed to play a central role in the motor control of organisms, given their extensive interconnection with motor cortex and the peripheral motor nervous system. It is recognized as a control system intended to minimize the error between the actual and predicted outcomes of motor actions, as well as to continuously update an internal model of the environment necessary for such predictions. It is involved in the tuning of fast and coordinated movements, and it is thought to be involved in the learning of more complex motor commands. The goal of the following work is to test if one can develop a computational method that mimics the learning rule proposed to train artificial neural networks. The aforementioned cerebellar learning mechanism is presented in the following subsection. However, it is first necessary to introduce the basic circuitry elements of the brain and their relations.

Neurons are the fundamental units of computation in a biological neural network. These are interconnected in intricate ways forming populations that give rise to complex patterns of activity. Neurons communicate through electrical and/or chemical signals, and the synapse is the intermediate element between different neurons that allows them to pass messages. Synaptic plasticity is the mechanism that controls the effectiveness of communications at this junction and it allows the brain to adapt to and encode new information. Plasticity can depress(potentiate) the strength of a synapse, and the outcome is a more(less) effective message passing between the two neurons involved. In a simplified way, these elements are found in artificial neural networks. These are composed of units (analogous to neuron) and a scalar number that represents the weight (analogous to synapse) of their connections. The goal of training is to modify these weights, so that the network learns to perform a desired task.

The main input of the cerebellar network is a mixture of sensory, contextual, and motor information, and the output sends signals to different motor systems. The learning proposed in the paper to be performed by the cerebellum relies on random perturbations of the activity of this structure's output neurons, and this affects the error: a quantity that indicates the difference between the output of the network and what it would be its desired response. The main cerebellar theory describes it, as one can clearly deduce from the previous considerations, as a supervised learning mechanism. Information about the error evolution is used to drive synaptic plasticity. Namely, variations that lower the error are retained[1], whereas the ones that increase the error are suppressed[2]. This learning rule corresponds to a stochastic

[1] synapses of neurons active at the same time of the perturbation are potentiated
[2] synapses of neurons active at the same time of the perturbation are depressed

gradient descent of the error function, as trials and errors drives the descent of the gradient. The evidences supporting the mechanism at play were obtained from experiments performed on in-vitro samples of slices of mice brains.

The final objective is to implement this mechanism to train recurrent neural networks (RNNs), a special class of models that allows to treat time-dependent inputs. They retain a certain type of memory because the source of the signal entering the network is a combination of current input, and information relative to its previous state. The characteristic that distinguish them and makes their application relevant, i.e. exhibition of temporal behaviour, is also the main obstacle to their training. Clearly, effects of a given modification can persist in the dynamics and show up only at later times.

In the chapter *Methods and Tools*, it is introduced the model of RNN adopted throughout the project (Section 2.1), and two techniques to train recurrent networks: *FORCE* (Sussillo and Abbott (2009)) and Node Perturbations (Fiete and Seung (2006)). The principles of their learning rule will be discussed, along with the necessary mathematical framework, and the limitations related to their biological plausibility. In *Result*, a model is trained on a simple decision task by means of a variation of Node Perturbations (Miconi (2017)), which includes computations that are more feasible to be carried out by the brain. The scope is to unravel the dynamics involved both in the task performance and in the learning process (Section 3.2 and Section 3.3). The final section (Section 3.4) presents an hybrid approach to generate arbitrary time-dependent patterns as output, inspired by the algorithm developed by Miconi.

2 Methods and Tools

2.1 Model description

Throughout the project, it is adopted a conventional class of models for recurrent neural networks. The model is composed of N units (*neurons*) which dynamics are defined by the set of coupled first-order differential equations (Sompolinsky et al. (1988)):

$$\tau \frac{d}{dt} x_i(t) = -x_i(t) + \sum_{j=1}^{N} J_{ij} r_j(t) \tag{2.1}$$

where $x_i(t)$ is the total input current of unit i, a continuous variable that represents the collection of signals received from the ones connected to it (*activity* or *excitation*). The nonlinear transfer function $r_i(t) = h(x_i(t))$ defines the input-output characteristic of the unit, where $h(x) = tanh(x)$ is chosen for the following study (*response* or *rate*). This can be interpreted in the biological context as the output electrical activity, for example in terms of firing rate of unit. The firing rate represents the frequency of signals which are transmitted by the neuron, however the function chosen can assume both positive and negative values which it is not realistic.
The recurrent connections of the network are described by matrix $\boldsymbol{J}$, and its elements $\{J_{ij}\}$ (*synaptic couplings*) are independently drawn from a Gaussian distribution with zero mean and variance g/N. Parameter g scales the strength of the connections, and it has been proven rigorously that networks with $g > 1$ exhibit spontaneous chaotic activity prior to training. The neural net is initialized in this regime for reasons that will be explained in the following section. The set of components and parameters described above define completely the model in use. The dynamics of a network composed of N neurons can be thought as a time-evolving trajectory in a N-dimensional space, whose each axis represents the value of the activity of a single neuron (however the neuronal activity may be represented in the computational model)

The class described has no specific purpose a priori, and it can be implemented to perform a large number of tasks[1]. To do so, it is necessary to define the network input(s), and output(s), and what the network is supposed to do/perform. The error is computed, then, as the difference between the desired and actual output of the network. The goal of training the network is to minimize this quantity by adjusting the synaptic couplings. Change in connections will result in a change in behavior, and those modifications that allow to decrease

[1] just few examples will be introduced in the following sections

the error are sustained. In the subsequent sections, two class of training procedures are presented.

2.2 FORCE

FORCE is a technique to train recurrent neural network in face of spontaneous chaotic activity. It consists in an effective modification of connections to organize the irregular activity into coherent outputs (Sussillo and Abbott (2009)). The output of the network is defined as a linear readout unit:

$$z(t) = \boldsymbol{w}^T \boldsymbol{r}(t) \tag{2.2}$$

where $\boldsymbol{w}$ is a vector of elements drawn from a uniform distribution on interval $[-1, 1]$, weighting the neurons rates $\boldsymbol{r}(t)$ (*external couplings*).

The paper shows that the model is able to learn to produce single complex signals, multiple signals onset by control inputs, and patterns resembling collected data on human motions, like running and walking. To perform a successful learning, it is necessary to tackle three main problems: the feedback of erroneous output signal into the network, the credit assignment, and the suppression of chaotic behaviour. A simple task, to which we can lead back more complex ones, is taken as a reference to discuss these three issue. The task consists in the network output to follow a precise time-dependent function, which is called target function.

The first problem may cause the network activity to diverge from the desired one, resulting in the inability of learning to converge. It was avoided in previous works (Jaeger and Haas (2004)) by setting the feedback signal equal to the target function to be learned, thus removing feedback errors: this class of RNNs is called Echo-State machine. However, it is hard to imagine how the brain may implement such function, as a replica of the target should be available for comparison. The strategy of *FORCE* algorithm to alleviate this possible impeachment of learning is also its principal characteristic. Namely, it is able to largely minimize the output error in the initial phase of learning, so that the signal fed back to the target is rather similar to the target function. The aim of the procedure then shifts to maintain this error small, but non-zero, to allow the sampling of the fluctuations of the system throughout the rest of the learning phase. This is proven to considerably increase the stability of the network, which is also a crucial weak point of recurrent nets.

The second problem, which is one of the crucial questions to answer in every artificial network training, is credit assignment: how to identify the elements that contribute the most to output error. Namely, where modifications in synaptic couplings, and as a consequence in neurons activity, are most needed to generate the correct pattern. The issue is particularly severe in systems with hidden states, that are the representations of previous inputs, as it is the case for recurrent nets. Units do not produce the output directly, and a objective function for the activity of each of them is not available a priori.
Modifications are only restricted to the external couplings $\boldsymbol{w}$ (Eq. 2.2) to alleviate the credit

assignment problem, following the approach of Jaeger and Haas (2004). This computational technique is named Reservoir Computing: the recurrent net functions at the edge-of-chaos, and it is able to produce a large repertoire of dynamical trajectories, which are selected to the appropriate needs by fine tuning of the external weights to combine the output signal. Furthermore, the application of *FORCE* can be extended to successfully train more complex architectures. For example, ones where both internal and external couplings are modified, and also ones where the feedback is provided by a separate recurrent network, whose connections to the primary network are subject to modification as well.

Echo-state machines do not suffer from issues related to chaotic behavior's characteristics: among these, one can mention the exponential sensitivity to initial conditions, or the display of complex activity. These are avoided by setting the initial state of the network inactive in absence of input. The problematic with this computational solution, as it is shown in the paper by Jaeger and Haas, is that it is detrimental to the training procedure. Networks characterized by initial chaotic dynamics benefits from this property: they require less steps to train and produce more robust outcomes that are less sensitive to noise, disturbances, and initialization.
A characteristic of electrical activity recordings in biological neuronal populations is to exhibit spontaneous irregular patterns, even in the absence of input stimuli. It is suggested that these dynamics play functional roles in the encoding of information, and may represent a local resource to explore the high-dimensional state space of the network (Mante et al. (2013)). The implementation of this property appears then crucial in an effort to model artificial networks in closer resemblance to natural ones. *FORCE* overcomes any difficulty related to initial chaotic activity by suppressing this dynamics at the onset of the training phase. It is obtained by a strong initial synaptic modification that generates controlled patterns of activity, close the ones desired. It is important to reiterate this concept, as it is the central mechanism allowing *FORCE* to produce successful results in training chaotic recurrent neural network, solving two out of the three main problems of the process.

In the paper, the mathematical framework of the learning rule and the reasons behind how the powerful synaptic modification of *FORCE* emerges from it are introduced in the following. To do so, a simple architecture is used (Figure 2.1), related to the concept of Reservoir Computing. The dynamics of the network are defined by the set of first order equations:

$$\tau \frac{d}{dt} x_i(t) = -x_i(t) + \sum_{j=1}^{N} J_{ij} r_j(t) + z(t) w_i^{FB} \tag{2.3}$$

where the output $z(t)$, a linear combination of neuronal rates (Eq. 2.2), is fed back to the system, through the vector of weights $\boldsymbol{w}^{FB}$, whose element are drawn from a uniform distribution on the interval $[-1, 1]$. This feedback signal is the sole source of input of the network. As anticipated previously, the general aim of training is to set the output equal to

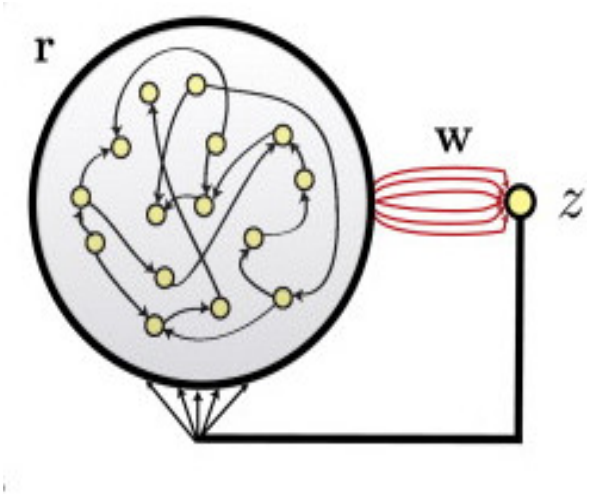


Figure 2.1: This is the architecture used by Sussillo and Abbott to introduce the concept of FORCE learning. In red, the connections that are modified by the procedure. The thick black line represents the output fed back to the network.

a given target function: $z(t) = f(t)$.
At step t, the vector of external weights is updated according to the recursive least-square (RLS) algorithm (Haykin (2002)):

$$\boldsymbol{w}(t) = \boldsymbol{w}(t - \Delta t) - e(t)\boldsymbol{P}(t)\boldsymbol{r}(t) \tag{2.4}$$

where Δt is the time-step of the simulation, and $e(t) = |z(t) - f(t)|$ is the output error. The pre-synaptic rates $\boldsymbol{r}$ are combined with information about output error in a standard delta-type rule to modify the couplings. The elements of the $N \times N$ matrix $\boldsymbol{P}$ are interpreted then as a set of multiple and adaptive learning rates. $\boldsymbol{P}$ is updated along with the weight vector:

$$\boldsymbol{P}(t) = \boldsymbol{P}(t - \Delta t) - \frac{\boldsymbol{P}(t - \Delta t)\boldsymbol{r}(t)\boldsymbol{r}^T(t)\boldsymbol{P}(t - \Delta t)}{1 + \boldsymbol{r}^T(t)\boldsymbol{P}(t - \Delta t)\boldsymbol{r}(t)} \tag{2.5}$$

$$\boldsymbol{P}(0) = \frac{\boldsymbol{I}}{\alpha}$$

where the initialization parameter α is a scalar number. It can be observed that the matrix $\boldsymbol{P}$ corresponds to a running estimate of the inverse correlation matrix of the rates of neural activity plus a regularization term $\boldsymbol{P} \approx (\sum_t r(t)r(t)^T + \alpha \boldsymbol{I})^{-1}$. It is immediate to illustrate the power of RLS for the purpose of training the network, according to the previous analysis of the main problems linked to it.
If we initialize the weight vector to $\mathbf{0}$, after the first update:

$$e(\Delta t) = -\frac{\alpha f(\Delta t)}{\alpha + \boldsymbol{r}^T(t)\boldsymbol{r}(t)} \tag{2.6}$$

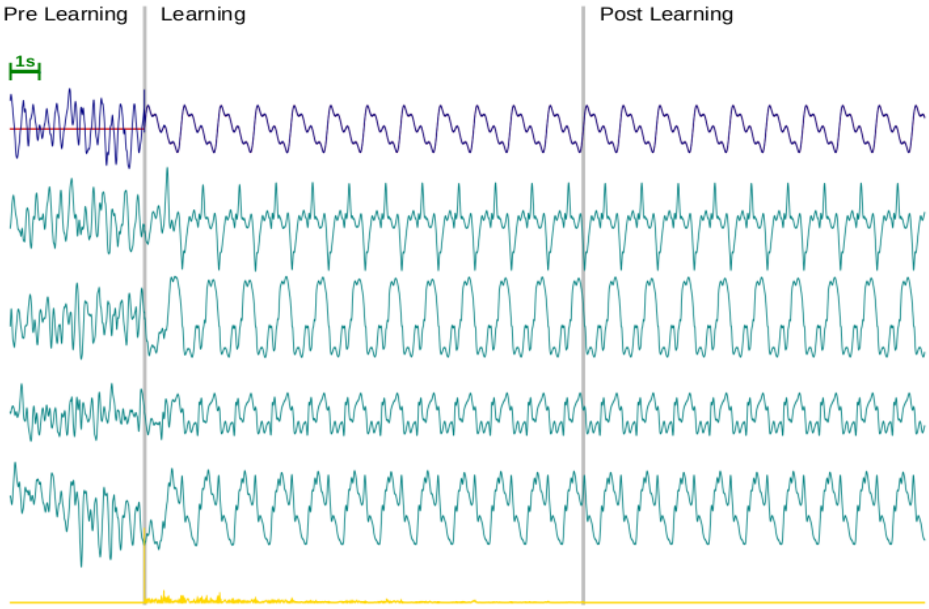


Figure 2.2: It shows the results of my implementation of the FORCE algorithm to train the network to follow a sum of simple sine waves. Dark blue and red lines represent respectively the output of the network and the target function: the error between output and target is small from the initial phase of training. Light blue lines represent the activity x of four neurons, and it is possible to observe how a transition from chaotic, spontaneous activity to regular patterns occurs at the onset of the procedure. Yellow line represents the norm of the modifications performed on the external weight vector $|\dot{\boldsymbol{w}}|$, the largest ones occurring in the first part of training. After the target is learned, the network is able to reproduce it spontaneously, without further modifications.

the error is small for parameter $\alpha \ll N$, as $\boldsymbol{r}^T(t)\boldsymbol{r}(t) \sim \mathcal{O}(N)$.

This encloses the salient feature of *FORCE*, which stands for First Order Reduced and Controlled Error: the ability to induce rapid and effective synaptic plasticity to minimize output error in the beginning of the procedure, i.e. firmly control the output to be close to target. This allows to shift the focus of the successive steps in maintaining the error small. In this way, the network is able to sample fluctuations and finely tune the weight vector, i.e. learn the static set of values that generates the desired patterns. Effectively, the training is divided in two phases: controlling, and learning. The former is fundamental to suppress the irregular activity during the process, a necessary characteristic that was discussed previously. Indeed, the feedback signal is close to the target function and it is able to drive the network in a non-chaotic regime (given it possesses the properties necessary, which are detailed in Rajan et al. (2010)). The latter assures not only that $f(t)$ is reproduced, but also that the system is stable. The success of this learning rule, however, comes with few limitations in the correspondence with biological implementations.

First of all, the architecture adopted above to describe the functioning of the procedure: the strong feedback pathway, which broadcast the same signal to all neurons, does not have a natural counterpart. It is thus necessary to separate it from the output unit, and two more

realistic architectures are proposed.

In the first, the feedback is generated by a secondary network where plasticity occurs also in its sparse connections to the 'generator' network. The second does not receive any feedback at all, however learning takes place on the internal synapses as well. The procedure is extended by treating any neuron, which synapses are being modified, as the readout unit. This to overcome the problem of credit assignment, as their effect is not direct on the output. The tests show that both are able to perform similar complex tasks to the ones that the non-realistic architecture does.
Rather than pretend these cases to explain functional aspects of brain networks, a more conservative approach would be to use them to model their activity. The loop structure of the first one is heavily implemented in neural systems, as the learning and modifications occurring in the loop does not disrupt the main network, which could have a drastic impact. An example of where this could be found in the brain is in the relation of motor cortex (generator) with cerebellum or basal ganglia (feedback loop), in learning basic and advanced movements.
The motivation to introduce these new architectures is to progressively remove non-biological elements in the *FORCE* procedure. However, an intrinsic feature of the algorithm does not meet natural constraints: its neuron-specificity, and not synapse-specificity. What this means is that the update of a synaptic coupling is performed yielding information on the activity of all neurons for the neuron involved. This can be observed as Δw_i contains a combination, through the product of matrix $\boldsymbol{P}$ and $\boldsymbol{r(t)}$, of the rates of all units. This is in stark contrast with the evidences collected on the mechanism of synaptic plasticity, which it is believed to be driven by solely local information available to the synapse. A possible solution would be to substitute $\boldsymbol{P}$ with a scalar quantity, although performances and the complexity of tasks executable are greatly diminished.

In the next section, the focus will be on a training method that is based on a completely different approach. It allows to have synapse-specific update of couplings, removing a central biologically implausible element of *FORCE*. Nonetheless, some biological limitations persist in this implementation as well. These will be analyzed, along progressive works that have been carried out in the literature and aim to obtain furthermore biologically plausible learning algorithms.

2.3 Node Perturbations

Node perturbation is a stochastic gradient descent (SGD) technique to train artificial neural networks. It was first introduced for feed-forward models (Widrow and Lehr (1990)), and later Fiete and Seung (2006) proposed a version for recurrent ones. Fiete and Seung's algorithm is model free, namely it was not derived for a specific recurrent architecture, thus, it can be implemented for the case described in Section 2.1. The stochastic element is the perturbation of network activity that it is suitably exploited to allow the descendant along an high-dimensional function, which represent the error of the network. The iterative process enables to find the (local) minimum of the function by travelling in the opposite direction of the gradient, optimizing the set of weights. The founding idea is to perturb the activity of single neurons and compare the generated output error against a baseline one, which has not received any perturbation. The difference between these two quantities is used to estimate the gradient of the objective function for learning. Moving along the gradient, the procedure is able to explore different local minima of the function. The ensuing mathematical framework proves rigorously how the algorithm performs the gradient descent and it is derived in Fiete and Seung (2006).

The system evolves for multiple trials of time-length T, generating along the way a trajectory $\Omega(t)$ at each trial, that characterizes the activity of the network. At the end of each, a functional of the trajectory $R[\Omega]$ is computed. This can be considered as a reward, in reinforcement learning terminology, for the given trial: it plays the role of the objective function for the learning, and it is an evaluation of the 'correctness' of the trial. A new dynamical variable is defined that represents the total post-synaptic activity of neuron i, namely its output:

$$g_i(t) = \sum_{j=1}^{N} J_{ij} x_j(t) \tag{2.7}$$

the pre-synaptic activities of the neurons connected to neuron i, weighted by the respective synaptic couplings. The linear dependence of g_i on synaptic couplings plays a crucial role in what follows. If one supposes that J_{ij} is a time-dependent function, it is possible to rewrite the infinitesimal variation of the reward with respect to the weight of the connections by simple chain rule:

$$\frac{\delta R}{\delta J_{ij}(t)} = \frac{\delta R}{\delta g_i(t)} x_j(t) \tag{2.8}$$

however, this is not the case as couplings have static values along a trial. Nonetheless, the technical issue can be disregarded, by the following mathematical trick:

$$\frac{\partial R}{\partial J_{ij}} \equiv \frac{1}{T} \int_0^T dt \frac{\delta R}{\delta J_{ij}(t)} = \frac{1}{T} \int_0^T dt \frac{\delta R}{\delta g_i(t)} x_j(t) \tag{2.9}$$

that makes it evident that dynamical variable g_i can be used, as a proxy, to perturb static weights J_{ij}.
First of all, a white noise term is applied to the activity:

$$g_i(t) = \sum_{j=1}^{N} J_{ij} x_j(t) + \xi_i(t) \tag{2.10}$$

which statistics are defined as $\langle \xi_i(t) \rangle = 0$ and $\langle \xi_i(t) \xi_j(t') \rangle = \sigma^2 \delta_{ij} \delta(t - t')$ (average is taken over a single trial). For small values of $\xi_i(t)$, it is possible to approximate R up to first order:

$$R - R_0 \approx \int_0^T dt \sum_{k=1}^{N} \frac{\delta R}{\delta g_k(t)} \xi_k(t) \tag{2.11}$$

where R_0 is the necessary noiseless baseline for the reward function, i.e. R in absence of any perturbation. This represents an important quantity, as its choice can greatly affect the convergence of the gradient descent procedure. In the following, it will be taken as a running average of the reward value. Given that noise is spatially and temporally uncorrelated, one can combine this information with the previous equations (Eq. 2.9 and 2.11):

$$\langle (R - R_0) \xi_i(t) \rangle \approx \sigma^2 \frac{\delta R}{\delta g_i(t)} \xi_i(t) \tag{2.12}$$

$$\sigma^2 \frac{\partial R}{\partial J_{ij}} \approx \frac{1}{T} \int_0^T dt \langle (R - R_0) \xi_i(t) \rangle x_j(t) \tag{2.13}$$

From this final result, it is intuitive to derive the stochastic learning rule. Over the trial, it is accumulated the so called eligibility trace:

$$e_{ij} = \int_0^T dt \xi_i(t) x_j(t) \tag{2.14}$$

that conveys information purely local to the synapse. At the end of the trial, the synaptic couplings are updated according to:

$$\Delta J_{ij} = \eta (R - R_0) e_{ij} \tag{2.15}$$

which is a stochastic variable due to presence of noise. However, on average $\langle \Delta J_{ij} \rangle \propto \partial R / \partial J_{ij}$, the learning rule follows the gradient of the reward function. As it was anticipated, *node perturbation* is proven to be a stochastic gradient descent method.

This procedure is adopted in the paper by Fiete, Fee, et al. (2007) to model song learning occurring in song birds, extending the computational scheme firstly proposed by Doya and Sejnowski (1998) with biological-supported data and evidences. The components of the general model are: actor, critic, and experimenter. The task, e.g. the song production, is car-

ried out by the actor and evaluated by the critic. It is thought that the critic owns a template of the target for comparison, e.g. in the juvenile bird this is learned from a senior individual. Learning occurs in a trial and error fashion: variations to the performances, produced by the experimenter, are reinforced if beneficial or hindered if detrimental. This capture the essence of *node perturbation*, and it highlights the resemblance to the case of cerebellar learning discussed in the introduction (Bouvier et al. (2018)). Despite the aforementioned case for songbirds, implementations of the algorithm in neural circuits does not seem evident, as the synapse should device a mechanism to keep separated the signal due to the external perturbation and the actual current input to produce the eligibility trace. Moreover, it is in contrast with the conventional model supported for synaptic plasticity, namely Hebbian theory. It suggests that the two forms of inputs multiplied to perform plasticity should be the pre- and post-synaptic activity (*neurons that fire together, wire together*, for a review Hebb (1949)), which is not the case occurring here. Two variations of *node perturbation* propose alternative learning rules that may be more plausible to be adopted in biological networks.

The first one (Legenstein et al. (2010), Hoerzer et al. (2012)) extracts perturbations from rapid fluctuations of neural output activity g. The algorithm allows network to reproduce periodic signals, perform task-specific computations, and the emergence of working memory is observed. The update of synaptic couplings consists of:

$$\Delta J_{ij} = \eta x_j(t)(g_i(t) - \overline{g}_i(t))(R(t) - R_0(t)) \tag{2.16}$$

where $\overline{g}_i$ and R_0 represent a fast running average of the output and the reward signal, respectively. The high-pass filtered version of the pre-synaptic activity substitutes the noise term in Eq. 2.14. The rule is in agreement with Hebbian theory, as update is based only on information locally available to synapses. Despite this step towards a more realistic mechanism, the presence of a continuous reward signal is in stark contrast with scientific evidences: only sparse rewards are provided in most reinforcement experiments or tasks. Unfortunately, the real-time signal is necessary to extract reliable perturbation, as it counters spurious relaxation effects arising from the subtraction of the fast running average (relaxations in both $g(t)$ and $R(t)$ cancel each other out).

To avoid this undesirable feature, Miconi (2017) proposes a change to the eligibility trace that stems directly from observations on spurious effects: perturbations have a larger magnitude than relaxations. Thus, a supralinear function applied the product of pre- and post-synaptic activity increments would diminish the negative influence of relaxations, and increment the relative weight of perturbations. To allow for exploratory behavior during training, the activity of neurons is perturbed directly, and not their putput activity. The dynamical

equation describing the evolution of activities becomes:

$$\tau \frac{d}{dt} x_i(t) = -x_i(t) + \sum_{j=1}^{N} J_{ij} r_j(t) + \sum_{j=1}^{N_{in}} W_{ij} u_j(t) + \Delta_i(t) \tag{2.17}$$

where perturbations are distributed according to a uniform distribution on interval $[-0.5, 0.5]$, spatially and temporally uncorrelated ($\langle \Delta_i(t) \Delta_j(t') \rangle = \sigma^2 \delta_{ij} \delta(t - t')$). The N_{in}-dimensional external input signal $\boldsymbol{u}$ is connected to the network through the weight matrix $\boldsymbol{W}$, which elements are drawn from a continuous uniform distribution on interval $[-1, 1]$. Eligibility trace is taken as:

$$e_{ij} = \int_0^T dt S(r_j(t)(x_i(t) - \overline{x}_i)) \tag{2.18}$$

where a running average $\overline{x}_i$ is subtracted from the activity of neuron x_i, and the rate of neuron j is taken as pre-synaptic activity. The crucial element is $S(x)$, a supralinear function of x (for example, in the paper, $S(x) = x^3$ is adopted). In this way, larger increments (that may carry informative correlations) are favoured over smaller ones (that may just be incidental). This may also be related to emerging theories of threshold-based Hebbian plasticity (Soltoggio and Steil (2013)), where there exists evidence for an implementation of plasticity occurring above a given value of combined electrical signals.
The eligibility trace accumulated along the trajectory for a single trial (Eq. 2.18) is combined with reward information to obtain the effective update to synapse couplings according to:

$$\Delta J_{ij} = \eta (R - R_0) e_{ij} \tag{2.19}$$

which is the same equation used in *node perturbation*. Sparse reward signal is restored, accompanied by Hebbian plasticity (since inputs and perturbations must no longer be segregated, as the former is directly applied to the activity). Hence, *Miconi's learning* avoids the major unrealistic elements of the earlier two procedures. The procedure is proven to be successful in learning to perform two cognitive tasks and to model data recorded on arm movements. Even so, a precise mathematical framework to explain why it functions is still lacking. This leaves open questions about the extent to which this algorithm can be applied to.

In the next chapter, it will be analyzed one of the cognitive tasks introduced in the paper. This to have a better understanding of how the network learn, and how it is able to encode information related to the specific task, in a dynamical way.

3 Results

3.1 Delayed nonmatch-to-sample task

It is relevant to dive into the details of the algorithm proposed by Miconi, given the affinity with the cerebellar stochastic perturbations mechanism igniting the current thesis work. The goal is to analyze and visualize how the dynamics of the network evolve due to the training procedure to accomplish the desired output. To do so, I have implemented the first task treated in its paper. It is a basic example to start with, which has been extensively adopted in animal experiments in cognitive neuroscience. Although its simplicity, it exhibits two important and absolutely non-trivial aspects of brain functioning: the network needs to retain information about its past activity (short-term memory) and it has to perform a non-linear categorization. To understand what one means by these two aspects, it is necessary to present what the task consists of.

The network gets two stimuli in input, spaced apart by a fixed temporal delay. The response must be equal to **-1** if the two stimuli are the same or **+1** if they are different. The first and second stimuli can take in two different signals (labelled as **A** or as **B**), thus, four possible combinations can be received by the network in a single trial: **AA**, **AB**, **BA**, or **BB**. The dynamics are then described by the same set of first order differential equations as Eq. 2.17. $\boldsymbol{W}$ is a $N \times 2$ matrix of external weights[1] to introduce the signal into the network, and the input assumes value $u_A = 1 \ \wedge \ u_B = 0$ if the stimulus is **A** or vice versa if the stimulus is **B**. This keeps the two input channels divided. A single trial has a length of $1000\ ms$, and it consists of three 'relevant' intervals of length $200\ ms$, in between which are present two 'silent' intervals acting as delay. In the first two 'relevant' intervals, the network receives the two stimuli, whereas in the last one it has to evaluate if they were the same or not, and this is why it is named *evaluation* interval. Note that the task corresponds to an exclusive-or problem with delay in time. The response is taken as the firing rate r of a randomly drawn neuron in the network, averaged over the evaluation interval: $\overline{r}_{out}$.

The learning consists in repeating the single trials over and over (I trained it on 10000 number of trials, but increasing the value up to 20000 or 30000 trials allows for better stabilization of the network). At each different trial, one of the four possible combinations is presented at random. Eligibility trace is accumulated along the trial, and at the end of it, the effective synaptic coupling modification is applied (Eq. 2.19), where reward and its baseline

[1] The elements are drawn from a continuous uniform distribution on the interval $[-1, 1]$

value are defined as:

$$\begin{aligned} R &= 1 - \frac{|1 - 2\delta_{ij} - \overline{r}_{out}|}{2}, \quad i, j = \{\mathbf{A}, \mathbf{B}\} \\ R_0 &= \alpha R + (1 - \alpha) R_0 \end{aligned} \tag{3.1}$$

Namely, R is the complementary of the relative error, which in turn is defined as the distance of the output from the target (either +1 or -1), and R_0 its running average. The parameters used to update synaptic couplings are $\eta = 0.3$ and $\alpha = 0.75$. The reward signal is produced at the end of each trial, thus is delayed and sparse in a more realistic fashion. Moreover, the weight modifications at the end of each trial cannot exceed an absolute value of 3×10^{-4} to avoid possible diverging increments, which would impair the stability of the learning. This could be related to the mechanism of saturated plasticity, dealt with in the paper by Nguyen-Vu et al. (2017).

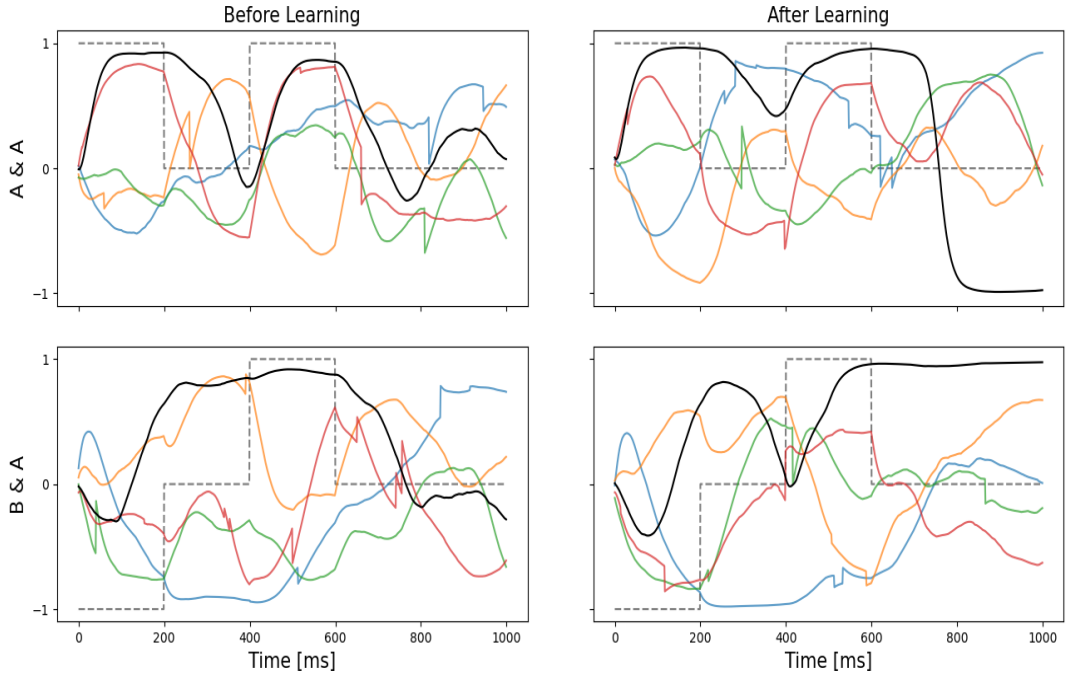


Figure 3.1: The black line represents the rate of the output neuron over the course of a single trial, whereas the colored ones the rates of randomly chosen neurons; the grey dashed line is a symbolic representation of the inputs received. On the left is displayed the network activity at the beginning of the training, and on the right after the training is completed

Another crucial element of the learning procedure, that allows the network to explore large portion of the state space, are the stochastic perturbations of neuronal activity $\Delta_i(t)$. These are applied independently to each neuron[2] with a mean frequency of $3\ Hz$ and an amplitude which is drawn from a continuous uniform distribution on the interval $[-0.5, 0.5]$. Their presence is evident in the jumps in activity displayed in Figure 3.1.

[2]Except to the neuron which is chosen as output of the network

One can observe from Figure 3.2 how the error steadily decreases in the first 3000 trials for all 4 combinations of stimuli. Even though some spurious fluctuations in the correct response of the network are still present after these, the overall performance of the network is highly accurate. These can be reduced, if one increases the number of total trials performed as stability is enhanced. After training, the model is tested over 1000 trials[3] and the average output error is $1.9 \pm 0.2\%$.

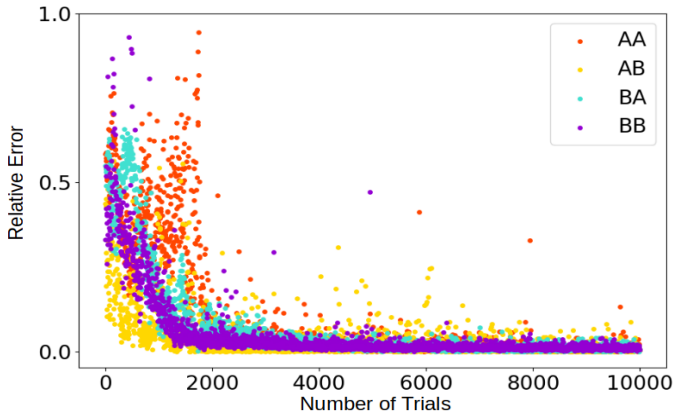


Figure 3.2: The graph displays the evolution of the error as a function of the number of trials (in this the total number is 10000). The dots are color-coded according to the combination of stimuli received by the network.

The activity is very dynamical, before and after the training is applied, as it is shown in both left and right columns of Figure 3.1. The only difference between the two is that the output rate clips during the evaluation interval to the correct value associated to the combinations of stimuli. It is indeed this same dynamical activity that allow the network to perform the cognitive task: the delay is larger than the time constant of the model $\tau = 30\ ms$, thus it needs to store temporarily the information about the first stimulus after it is over, and manipulate it actively once the second one is also received, to respond properly. This is the basic characteristic of a working memory tasks. The behavior observed here is closely related to findings presented on information encoding in prefrontal cortex by neural activity (Barak et al. (2010), Stokes et al. (2013)). They show how patterns of single neurons are highly dynamical, observed to fluctuate from trial to trial, in response to the incoming stimuli.

The activity of neurons can be analyzed with statistical tools to shed insights on their representations of signals. This is the goal of the next subsection. As well as to show findings that are in accordance with observations in cortical recordings and the model proposed along with them.

[3]The combinations of stimuli are presented randomly

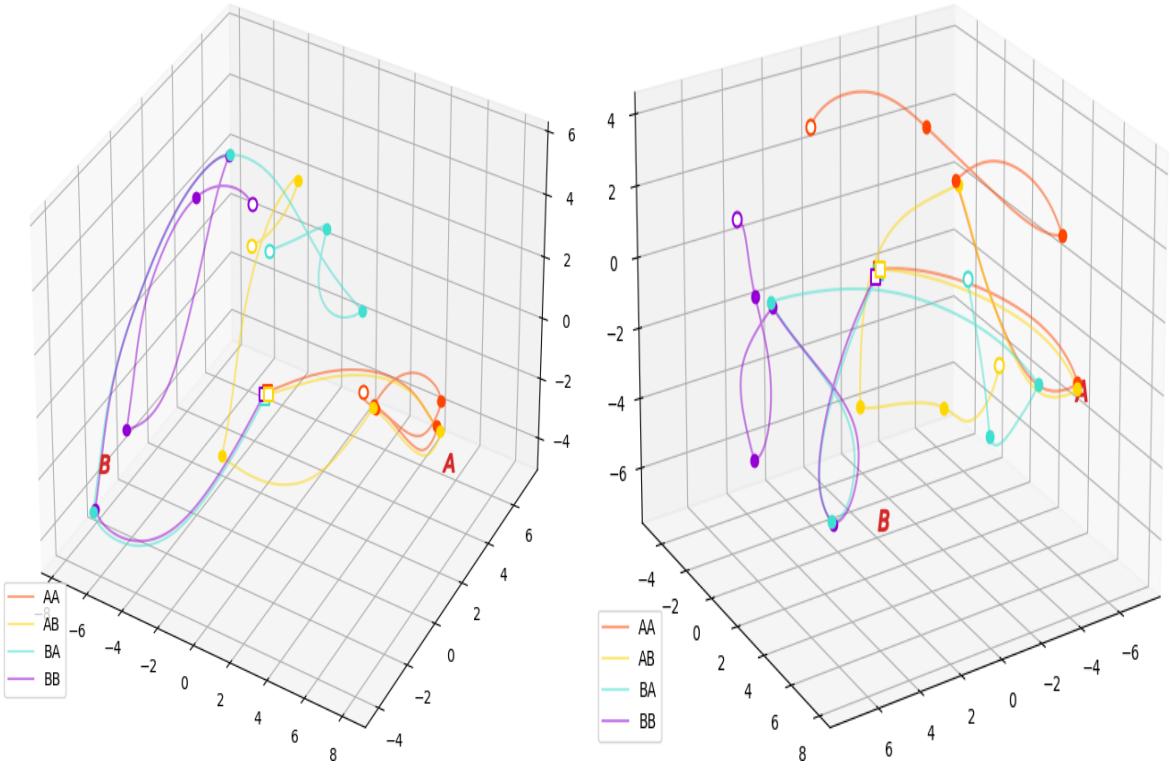
3.2 Low-dimensional activity

The network used for the whole project is composed of $N = 200$ neurons. The state of its dynamics evolves in time, then, in a vast N-dimensional space. Even though it displays complex and highly variable patterns, it is hypothesized a priori that the activity, during the cognitive task presented before, actually lies in a manifold of dimensionality $n \ll N$. This since a low-dimensional space is sufficient to carry information about the task to be performed. This is observed also in recurrent neural networks performing task of temporal signals integration (Fanthomme and Monasson (2020)).
The analysis of the network is carried out by means of Principal Components Analysis (PCA), a statistical tool widely adopted to perform dimensionality reduction: its purpose is to describe a dataset which contains a large number of variables with a few number of variables, while retaining as much as possible the information present in the dataset. Geometrically, it individuates the directions where the amount of variance is maximal, which are the ones containing the relevant information of the data. These are called principal components (PCs) and are obtained from the eigenvectors of the covariance matrix of the dataset. It allows for visualization of what it is going on in the network during the performance of the task, as we can plot the projection of the dataset onto the first two or three PCs, and extrapolate useful insights, if the amount of variance explained is significant.

The analysis is extended to the study of different cases in which the delay assumes larger values than the standard $200\ ms$ one. The motivation behind this is to observe the network activity in between the two stimuli, as the crucial step in this working memory task. In this direction, one wants to understand how the phase space defined by the dynamical equations of the activity is related to the encoding of information, and possibly how it evolves in time as the time-dependent input modifies it. The delays considered are evenly spaced by $50\ ms$ in the interval $[200\ ms, 450\ ms]$. It is noted that above $450\ ms$ the learning procedure starts to lose stability and the average error output steadily increase with delay increments, above an accuracy level of 5%.

The dataset is a collection of the traces of single neurons along the duration of the whole trial for each single combination of stimuli, thus four matrices of dimension $N \times$ Trial Length. The four of them are concatenated together so that the total dataset, on which PCA is then performed, contains information about the activity of the network for all possible combinations of inputs. Then, the elements of the covariance matrix will represent the correlations between activities of different neurons at different time steps, regardless of the kind of signal received. This is necessary to project the single datasets on a low-dimensional space which is common to the four combinations.

The projections of single trials are plotted in the three-dimensional space of the principal components in Figure 3.3. Each point of the trajectory represent an instant in time along



((a)) Delay = 200 ms **((b))** Delay = 300 ms

Figure 3.3: Colored trajectories represent the time-evolution of the network projected on the reduced space of the first three principal components (PCs) of the activity. It is color-coded for the four different combinations of stimuli. The beginning of the trail is the empty large square symbol, its end is the large empty round symbol. Small empty round symbols along the trajectory represent in order: the end of the first stimulus, the beginning of the second stimulus, its end, and the beginning of the evaluation trial. The red letters represent the fixed point associated to the input **A** and **B**. In the left column (a) the delay between the successive stimuli is 200 ms, whereas in the second (b) it is 300 ms. The cumulative variance ratio explained by the first 3 PCs is: 74.7% for the $200\ ms$ delay, and 79.2% for the $300\ ms$ delay.

the trial. These points are new variables, linear combinations of the initial ones, thus they do not have an actual meaning. Nonetheless, they are useful for the interpretation of the evolution of the network activity. Initial points (square empty marker) are not identical for each set of stimuli, as network activity is initialized randomly, drawn from a continuous uniform distribution on the interval $[-0.1, 0.1]$.

Then, it is observed how combinations **AA** & **AB**, or **BA** & **BB**, evolve in the same direction, as the network received the same stimulus. Towards the respective fixed point of the phase space, under either stimulus **A** or **B**. Letting evolve the trained network with a constant input over a trial of large duration (over 10000 steps), the state of the network converges to a point that does not change in time: by definition the fixed point of the first order differential equations. In this way, I was able to identify the attractors of the two inputs, labelled by a red capital letter in the figure, and project them in the PCs low-dimensional space.

Once the first stimulus ceases to be received, the network starts to relax and it changes directions, given that the equations describing the dynamics have changed. In this interval, the information is encoded by the network within its trajectory. Then, the evolution of the two couples diverges, as it is expected given they received two different stimuli. However, the common feature is that directions undertaken by each of the four patterns are towards the fixed points associated to the respective second stimulus. The second interval of relaxation[4] is of particular interest. Namely, resemblances in the activity of the network receiving a combination of stimuli that elicit the same output response (**AA** & **BB**, or **AB** & **BA**) can be identified. The trajectories for same stimuli inputs relax back towards the region occupied at the onset of the second stimulus, thus each in a very different region with respect to the other, even though their response is common. Whereas the ones for different stimuli inputs converge towards a common region, although their starting points, i.e. at the end of the second stimuli, are well-distanced.
It is remarkable that these are observed regardless of the length of the delay between the two stimuli (it is observable in both plots in Figure 3.3, as also in the other cases treated which are not shown to avoid excessive plotting). This highlights how, during the task, the network does not simply store the first stimulus received, but actively manipulate it to encode the relative information.

The mechanism identified above for the two same-response couple of pattern inputs suggests that the information is encoded by the trajectory of the activity, rather than by the presence of attractors. The latter mechanism has been used to model brain activity during working memory task (Amit and Brunel (1997)). In these models, neural dynamics are lead to different stable positions based on the different events experienced (in our case the combinations of stimuli). This has the disadvantage that when the dynamics reach the fixed point, all the information about time is lost. In another category of models, the activity keeps varying with time in a low-dimensional subspace. This mechanism is able to provide stable ways to store memories, and it has been adopted to model motor cortex recordings (Inagaki et al. (2018)), and prefrontal recordings (Murray et al. (2017)). Given these remark, it is reasonable to hypothesize that the network obtained by training drives the activity to specific regions of slow dynamics for the two classes of response. Namely, in the evaluation interval the first order equation for the output neuron becomes:

$$x_{out} \approx \sum_{j=1}^{N} J_{out,j} x_j \tag{3.2}$$

where perturbation term is absent, since it is necessary for exploratory behavior, and input term as well, since it is null. In support to this, I computed the correlations between vec-

[4] It represents the delay between the end of the second stimulus and the beginning of the evaluation trial

tor $\boldsymbol{J_{out}}$, which represents the synaptic couplings connected to the output neuron, and the average activity of the network in response to the different input patterns in the evaluation phase $\langle \boldsymbol{r} \rangle_{eval}$. As it is evident from Table 3.1, the network activity is correlated with $\boldsymbol{J_{out}}$ in case of mismatching stimuli, and it is uncorrelated in case of matching stimuli.

	A	B
A	-1.62	1.74
B	1.82	-1.63

Table 3.1: It represents the correlation matrix of the weights of synapses connected to the output neuron with the average activity of the single neurons over the evaluation interval for the four input patterns. The elements are an average over the different cases of delay.

3.3 Low-rank connectivity

The spirit of the section is to shed light on the black box that is the network. It is able to learn how to perform a desired cognitive task, however many questions arise spontaneously on how it is able to do so. Where is information stored? How is it manipulated? Or what is the role of the connectivity? To grasp an idea on what it is actually going on, it is necessary to analyze the two main elements of the model: the activity of the neurons and the synaptic couplings between them. The former is treated in the previous subsection, whereas now the focus necessarily shifts to the connectivity matrix.

Randomly connected recurrent networks can only implement simple input-output transformations, as their activity is highly-dimensional (Rajan et al. (2010)). On the other hand, classical models rely on a structured connectivity (Wang (2002), Williamson et al. (2016)). Namely, neurons are clustered together to specifically encode a given stimulus or implement a given response in the task. A mixture of these two phases, chaotic and structured, appears to be the one that distinguishes cortical connectivity and activity (Mante et al. (2013), Harris and Mrsic-Flogel (2013)). One could investigate if similar characteristics emerge in the connectivity structure obtained at the end of the training. This final matrix can be decomposed in two terms:

$$\boldsymbol{J}_{final} = \boldsymbol{J}_{initial} + \Delta\boldsymbol{J} \tag{3.3}$$

The initial matrix $\boldsymbol{J}_{initial}$ is sampled with a given statistics, and it displays highly-dimensional, and irregular patterns of activity. Then, it is the synaptic update $\Delta\boldsymbol{J}$ performed by the procedure that presumably accounts for the structured part. An analogous structure is studied in Mastrogiuseppe and Ostojic (2018): the subject of study is a class of networks where the total matrix is the sum of a random matrix and a low-rank one. The motivation comes from observations on multiple, independent implementations of training recurrent network scheme (Jaeger and Haas (2004), Sussillo and Abbott (2009), Boerlin et al. (2013)). Each is developed for different objectives and purposes, however a similar solution is obtained: the learning amounts to a low-dimensional update of couplings, rather than a specific element-by-element one, which mathematically corresponds to a low-rank matrix modification. This is observed in the network trained by Miconi's method as well. To show this, $\Delta\boldsymbol{J}$ is factorized by eigendecomposition, so that it is possible to obtain a rank-n matrix version of the original one by product of the first n eigenvectors:

$$\Delta J^n_{ij} = \sum_{k=1}^{n} \lambda^{(k)} m_i^{(k)} n_j^{(k)} \tag{3.4}$$

where $\boldsymbol{m}^{(i)}$, $\boldsymbol{n}^{(i)}$ are the left and right i-th eigenvectors, and $\lambda^{(i)}$ the i-th eigenvalue. This rank-n matrix is added to $\boldsymbol{J}_{initial}$ and the task is tested on this 'new' connectivity.

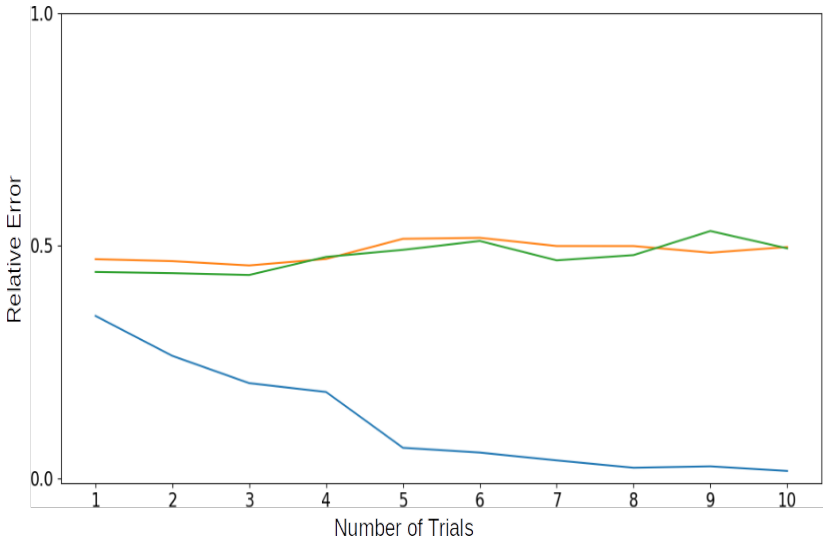


Figure 3.4: It represents the average relative error as a function of the rank of different parts of the connectivity matrix, tested on 1000 trials. In blue, the error of the task performed where the connectivity matrix is the initial connectivity $\boldsymbol{J}_{initial}$ plus the low-rank modification matrix $\Delta\boldsymbol{J}$. In green, the full trained matrix $\boldsymbol{J}_{final}$ is low-rank. In orange, $\boldsymbol{J}_{initial}$ is substituted by a new matrix sampled according to the same statistics. The traces give information about three crucial aspects and observations on the connectivity matrix, in order: that the modifications are minimal, that the chaotic part is necessary for the task performance, that a strong correlation between low-rank modifications and initial connectivity is present.

The results are collected in Figure 3.4. Blue trace represents the average relative error across 1000 trials for values of n from 1 to 10. For $n^* = 5$, the network is able to perform the task accurately, with a precision of 89%. This confirms the low-rank nature of modifications. The same procedure is extended to the total connectivity matrix $\boldsymbol{J}_{final} = \boldsymbol{J}_{initial} + \Delta\boldsymbol{J}$. This highlights the importance of the contribution from the underlying chaotic part to the success of the task, as the final connectivity does not display a low-rank nature. Moreover, I tried to shuffle the initial connectivity, by sampling a new one from the given distribution, add the low-rank modification, and perform the task. The orange trace represents the average relative error of these task performances. The network is not accurate,as its response is at chance level. This indicates a strong correlation between the chaotic and the low-rank component. The study of Mastrogiuseppe and Ostojic (2018) relies on the principal assumption that the two term are uncorrelated, to derive rigorous equations for the dynamics of the network. At the moment, a theoretical framework to take into account this case is still lacking.

3.4 Learn time-dependent target functions

The final part of the project is focused on the ability of the learning rule proposed by Miconi to produce complex time-dependent signals. This task is shown to be performed accurately by networks trained with FORCE algorithm, both in the paper and in my reproduction of results (Figure 2.2). The main limitation to the variation of node perturbation is a lack of a theoretical framework, even though it has been proven successful in training networks to perform multiple tasks. The absence of a justification of why the procedure results in a successful training poses serious questions on its extension to more complex tasks. Given these remarks, the objective of the following work is not to provide such framework, but to test the level of complexity that the algorithm can obtain.

The procedure implemented is a mixture of the two algorithms. The learning proceeds over a single long trial, instead of multiple smaller ones. The evolution runs over a total of $5000\ s$ in this case, and at intervals of $100\ ms$ the error is computed and the effective synaptic modification is applied. Thus, it leaves the reward signal distributed still in a sparse and delayed fashion. Equation 2.15 is applied to update the connectivity matrix. The initial time-dependent signal adopted to test our procedure is a simple sine wave function, with a frequency of $2\ Hz$, so that the eligibility trace would contain, at the end of its accumulation, the information about two cycles of the function. It was also used to test the intervals of the set of parameters that would yield the best result, i.e. the lowest error. The four main parameters that influence the learning of the trace are: the learning rate η, the time constant τ, the frequency of perturbations f_P, and the amplitude of perturbations A_P (the optimal intervals are referenced in Table 3.2, as learning beyond these is impaired).

η	0.1 - 0.7
τ	10 - 20 ms
f_P	15 - 30 Hz
A_P	10 - 20

Table 3.2: Optimal intervals for learning procedure of the parameters

The learning rule is further modified, taking inspiration from the learning mechanism described in Bouvier et al. (2018). The founding idea is that perturbations of neural activity resulting in a diminished(increased) global error should lead to a potentiation(depression) of such activity in the direction of the perturbation. In the implementation, perturbations are applied independently to all neurons with a given mean frequency. If a given neuron i at time step t would receive a perturbation, at the next time step $t+1$ its effect would be taken into account in the accumulation of the eligibility trace as follows:

$$e_{ij}(t+1) = e_{ij}(t) + \mu S(r_j(t) \cdot x_i(t))(\xi - \xi_0) \tag{3.5}$$

where ξ, ξ_0 are the error and its running average, this to determine if the perturbation is beneficial or less, and $\mu = 50$ is a secondary learning rate. This results in an optimization

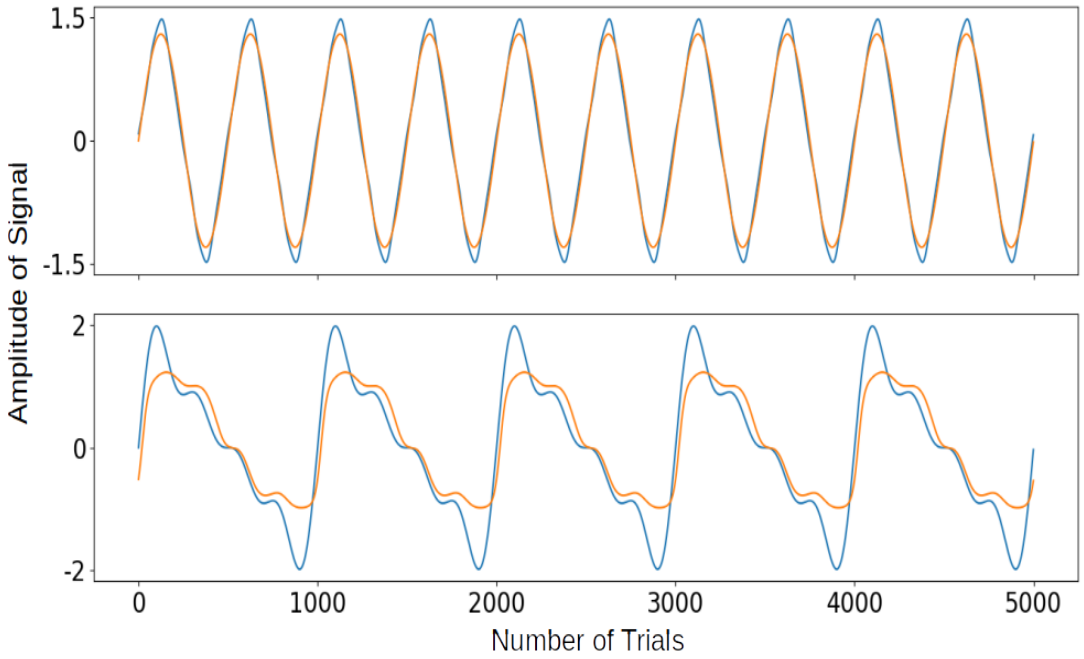


Figure 3.5: In the two panels are plotted the target function (blue) and the output of the network after the learning (orange). The upper panel is a simple sine wave, whereas the lower one is a combination of sine waves. It is clear how the learning of the amplitude of the signal is problematic for the network. The unit along the x-axis is milliseconds.

of the supralinear function, integrating the information about the change in error due to perturbation into the accumulated eligibility trace.

The result obtained are satisfying (upper image in Figure 3.5): the network is able to learn the frequency of the target function, however it displays a problem in learning the amplitude. This could be related to the fact that the exploratory perturbations heavily disturb the output of the network during learning, unable to produce a smooth function (this can be deduced from the evolution of the error, which oscillates around a non-negligible constant value that the learning is not able to reduce further). Indeed, it is only in the test phase, where perturbations cease to be applied, that one is able to observe that the network has partially learned the desired signal. The potential to learn simple periodic functions is present, even though further corrections are needed in order to resolve amplitude-related issues. Deviations from the function is accentuated in the second signal that the network was trained to reproduce: a combination of simple sine waves. Also in this case, the frequency-related learning is accomplished in large part, according to the frequency spectra of the output and target function.

The learning rule proposed by Miconi's shows some major limitations in the possibility of learning complex time-dependent signals, and this casts some doubts to the extent that it could be applied to. Moreover, a central element is the necessity to feedback the target function into the network throughout the procedure. This is obtained imposing the activity of 3 neurons to the exact copy of the signal, and given the full connectivity of the network, this information is broadcast to all neurons. It is fundamental as without it, the network is unable to learn or produce any periodic dynamics (the error evolution fluctuates around the chance value). This aspect is not plausible at the level of biological implementation, as explained in previous sections.

4 Conclusion

The algorithm proposed by Miconi is able to implement a cognitive task with great accuracy. It is characterized by a working memory aspect, thus, information must be stored and manipulated to execute the response correctly. It is plausible that the network encodes the incoming stimuli in a dynamical way. This is what is observed from the analysis on the low-dimensional trajectories of the network: the evolution of the system lies in a manifold whose dimensionality is much lower than the one of the state space, and, depending on the signals received, it is driven toward two regions of slow dynamics (specified by the matching/non-matching nature of the signal). These two well-separated regions allow for the network to respond accordingly, correlating the activity of the network with the weights of connections of the output neuron. This is important in the pursuit of network that could perform computations in a brain-inspired manner. This low-dimensional activity, that spontaneously arise from learning, has been observed as well in neural dynamics (Mante et al. (2013)). In addition, the analysis of the rank of the weight update performed by the procedure suggests a minimal structure modifications of synaptic couplings that remarkably transits the network activity from chaotic, and highly-irregular to low-dimensional, and organized. It is the same case for various other training procedures, therefore, it would be informative to study the possible presence in biological neural network of this efficient way to perform plasticity, which avoids a costly modification of specific synapses. Also, a rigorous treatment on how the two kinds of connectivity are related and influence one another would be important, and it is currently missing.

Although training the network on the delayed nonmatch-to-sample task -or other simple ones implemented in Miconi (2017)- results in accurate performances, the extent to which this algorithm can be brought is uncertain. The learning rule proposed has a strong inspiration from biological findings, however a theoretical framework that explains why the rule functions is still lacking (compared to the one provided for the original *Node Perturbation* in Fiete and Seung (2006)). This would allow to determine the extent of task difficulty to which the procedure can be applied to.

Regarding my further development of the variation to *Node Perturbation*, the results are encouraging even if not completely satisfying. The network is trained to output a trace that follows time-dependent periodic target functions (of increasing complexity). It is able to sample their frequency, however it exhibits issues in reproducing their amplitudes, a problem that is more evident as the complexity of the functions increases. This highlights already some limits of Miconi's algorithm as well, given the close resemblance between the two.

The problem is related especially to the presence of stochastic perturbations. These are the central mechanism that allow the training to be carried out, nonetheless the output of the network when they are present is strongly perturbed and it is not able to produce a smooth, steady function. It is only at the end of the process that the final result is obtained: a smooth function similar to the target one. This makes the procedure of extracting the error tricky, as does not necessarily represent the state of the network. To approach this problem, it would be necessary to find an optimal combination of the two parameters, amplitude and frequency of the perturbation, or a adapting procedure to vary them.
It is reasonable that further study in this direction will help overcome the problematic of learning in a noisy environment. Anyway, two major limitations to the applicability of the method still persist: the length of the training, and the feedback signal to the network. The former is due to the presence of costly matrix computations at each step to collect the eligibility trace used to update synaptic couplings. These make the running time of the algorithm 5 times slower than *FORCE*. It could be alleviated by GPU computations, but its performances will remain sub-optimal respect to the others. The second one consists in fixing 3 neuron activities to display the same desired time-dependent signal. Without this feature, and the information it carries, it is impossible for the network to converge to the correct response during learning. As it is explained in the second chapter, the presence of a mechanism to eliminate feedback error (or a structure that possesses a priori the target function to be input to the network) is highly unrealistic to be found in natural networks. Further developments will be necessary to move forward to learning tasks with a biologically inspired method.

The project has brought up different aspects and questions about the network ability to learn a task. From the study of the dynamics, single-unit activities, or connection weights, it is obvious how recurrent neural network are characterized by complex and intricate relations. These are puzzling to unravel and, to this day, accurate methods to pursue this objective are still limited. Contributions to this line of research would be a fundamental step forward to aid the extrapolation of useful information on actual neural dynamics. Artificial networks are extensively used as toy models for neural activity recordings. Thus, techniques to allow for a better understanding of the dynamics of the network combined with the development of more realistic training procedures can be brought together to possibly reveal -or at least provide hypothesis and test-beds- implementations of learning mechanisms hidden in human brain. In particular, modern technologies are improving further and further the level of resolution of recordings of increasingly larger populations of neurons: This gives unprecedented access to a large amount of detailed data on neural dynamics, and theoretical frameworks would be necessary to better capture the latent patterns. It is in this niche, that the efforts of my project were focused on.

Acknowledgements

Looking back on the academic adventure I have embarked two years ago, I have a great sense of pride and gratitude. I know the same is true for my family, who has been next to me along this path every single day. They have been kind enough to provide me with all their support, affection, and comprehension I definitely needed throughout even the most difficult times.

I want to send a special thank you to my mom above everyone, who gave me the strength to move always forward in my life with her beautiful example of sacrifice and determination, as well as the serenity to focus entirely on my studies. I want to thank my Grandmother, for her wisdom in living life fully, my Father and my Brother, for teaching me to enjoy the fundamental aspects of human interactions.

I am beyond amazed by the luck to have always encountered amazing people around me, each one gave its contribution to make me the person I am today. I am blessed to have a multitude of friends with whom I share the dearest memories: my second family Questa è Melta, my companions from high school (Pave and Bru), from the bachelor degree in Turin (Ema, Ste, Giuse, and many others), and the ones I had the great pleasure to meet along this journey (Ali, Albe, Costa, Fili, Fra, Luca, and Save). A sincere appreciation goes to professor Vincent Hakim, for his precious understanding, advise, and support, in the face of the complicated sanitary situation we had to carry out the internship in.

To all my family and friends, I dedicate this accomplishment. I would not be the person I am today without you. I promise to do my best to keep you be proud of me.

Mattia Della Vecchia

Bibliography

Amit, D.J. and N. Brunel (1997). "Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex". In: *Cereb Cortex* 3, pp. 237–252. DOI: 10.1093/cercor/7.3.237.

Barak, O., M. Tsodyks, and R. Romo (2010). "Neuronal Population Coding of Parametric Working Memory". In: *Journal of Neuroscience* 30.28, pp. 9424–9430. DOI: 10.1523/JNEUROSCI.1875-10.2010.

Blundell, C. et al. (2016). *Model-Free Episodic Control*. arXiv: 1606.04460 [stat.ML].

Boerlin, M., C. K. Machens, and S. Denève (Nov. 2013). "Predictive Coding of Dynamical Variables in Balanced Spiking Networks". In: *PLOS Computational Biology* 9, pp. 1–16. DOI: 10.1371/journal.pcbi.1003258. URL: https://doi.org/10.1371/journal.pcbi.1003258.

Bouvier, G. et al. (2018). "Cerebellar Learning Using Perturbations". In: *eLife*. DOI: 10.7554/eLife.31599.

Doya, K. and T.J. Sejnowski (1998). "A Computational Model of Birdsong Learning by Auditory Experience and Auditory Feedback". In: *Central Auditory Processing and Neural Modeling*. Ed. by MA Springer Boston. Poon P.W.F., Brugge J.F. DOI: 10.1007/978-1-4615-5351-9_8.

Fanthomme, A. and R. Monasson (2020). *Low-Dimensional Manifolds Support Multiplexed Integrations in Recurrent Neural Networks*. arXiv: 2011.10435 [cs.LG].

Fiete, I. R., Michale S. Fee, and H. Sebastian Seung (2007). "Model of Birdsong Learning Based on Gradient Estimation by Dynamic Perturbation of Neural Conductances". In: *Journal of Neurophysiology* 98.4, pp. 2038–2057. DOI: 10.1152/jn.01311.2006.

Fiete, I. R. and H. S. Seung (2006). "Gradient Learning in Spiking Neural Networks by Dynamic Perturbation of Conductances". In: *Phys. Rev. Lett.* 97 (4), p. 048104. DOI: 10.1103/PhysRevLett.97.048104.

Gregor, K. et al. (2015). *DRAW: A Recurrent Neural Network For Image Generation*. arXiv: 1502.04623 [cs.CV].

Harris, K. D. and T. D. Mrsic-Flogel (2013). "Cortical connectivity and sensory coding". In: *Nature* 503 (7474), pp. 51–58. DOI: 10.1038/nature12654.

Hassabis, D. et al. (2017). "Neuroscience-Inspired Artificial Intelligence". In: *Neuron* 95.2, pp. 245–258. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2017.06.011.

Haykin, S. S. (2002). "Adaptive filter theory". In: Upper Saddle River, N.J: Prentice Hall.

Hebb, D.O. (1949). *The Organization of Behavior, Wiley: New York; 1949*. John Wiley Sons. DOI: 10.4324/9781410612403.

Hoerzer, G. M., R. Legenstein, and W. Maass (2012). "Emergence of Complex Computational Structures From Chaotic Neural Networks Through Reward-Modulated Hebbian Learning". In: *Cerebral Cortex* 24.3, pp. 677–690. DOI: 10.1093/cercor/bhs348.

Inagaki, H. K. et al. (2018). "Low-Dimensional and Monotonic Preparatory Activity in Mouse Anterior Lateral Motor Cortex". In: *Journal of Neuroscience* 38.17, pp. 4163–4185. DOI: 10.1523/JNEUROSCI.3152-17.2018.

Jaeger, H. and H. Haas (2004). "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication". In: *Science* 304.5667, pp. 78–80. DOI: 10.1126/science.1091277.

Kirkpatrick, J. et al. (2017). "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* 114.13, pp. 3521–3526. DOI: 10.1073/pnas.1611835114.

Legenstein, R. et al. (2010). "A Reward-Modulated Hebbian Learning Rule Can Explain Experimentally Observed Network Reorganization in a Brain Control Task". In: *Journal of Neuroscience* 30.25, pp. 8400–8410. DOI: 10.1523/JNEUROSCI.4284-09.2010.

Mante, V. et al. (2013). "Context-dependent computation by recurrent dynamics in prefrontal cortex". In: *Nature* 503 (7474), pp. 78–84. DOI: 10.1038/nature12742.

Mastrogiuseppe, F. and S. Ostojic (2018). "Linking Connectivity, Dynamics, and Computations in Low-Rank Recurrent Neural Networks". In: *Neuron* 99.3, pp. 609–623. DOI: 10.1016/j.neuron.2018.07.003.

McCulloch, W. and W. Pitts (1943). "A logical calculus of ideas immanent in nervous activity". In: *Bulletin of Mathematical Biophysics* 5, pp. 115–133. DOI: 10.1007/BF02478259.

Miconi, T. (2017). "Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks". In: *eLIFE*. DOI: 10.7554/eLife.20899.

Murray, J. D. et al. (2017). "Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex". In: *Proceedings of the National Academy of Sciences* 114.2, pp. 394–399. DOI: 10.1073/pnas.1619449114.

Nguyen-Vu, T. B. et al. (2017). "A saturation hypothesis to explain both enhanced and impaired learning with enhanced plasticity". In: *eLife* 6. DOI: 10.7554/eLife.20147.

Rajan, K., L. F. Abbott, and H. Sompolinsky (2010). "Stimulus-dependent suppression of chaos in recurrent neural networks". In: *Phys. Rev. E* 82 (1). DOI: 10.1103/PhysRevE.82.011903.

Reed, S. E. et al. (2015). "Deep Visual Analogy-Making". In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2015/file/e07413354875be01a996dc560274708e-Paper.pdf.

Rosenblatt, F. (1958). “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65.6, pp. 386–408. DOI: 10.1037/h0042519.

Rumelhart, D.E., J.L. McClelland, and P.R. Group (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Volume 1*.

Soltoggio, A. and J. J. Steil (2013). “Solving the Distal Reward Problem with Rare Correlations”. In: *Neural Computation* 25 (4). DOI: 10.1162/NECO_a_00419.

Sompolinsky, H., A. Crisanti, and H. J. Sommers (1988). “Chaos in Random Neural Networks”. In: *Phys. Rev. Lett.* 61 (3), pp. 259–262. DOI: 10.1103/PhysRevLett.61.259.

Stokes, M. G. et al. (2013). “Dynamic Coding for Cognitive Control in Prefrontal Cortex”. In: *Neuron* 78.2, pp. 364–375. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2013.01.039.

Sussillo, D. and L.F. Abbott (2009). “Generating Coherent Patterns of Activity from Chaotic Neural Networks”. In: *Neuron* 63.4, pp. 544–557. DOI: 10.1016/j.neuron.2009.07.018.

Wang, X. (2002). “Probabilistic Decision Making by Slow Reverberation in Cortical Circuits”. In: *Neuron* 36.5, pp. 955–968. ISSN: 0896-6273. DOI: 10.1016/S0896-6273(02)01092-9.

Weston, J., S. Chopra, and A. Bordes (2015). *Memory Networks*. arXiv: 1410.3916 [cs.AI].

Widrow, B. and M. Lehr (1990). “30 Years of Adaptive Neural Networks: Perceptron Madaline, and Backpropagation”. In: *Proc. IEEE* 78 (1415).

Williamson, R. C. et al. (Dec. 2016). “Scaling Properties of Dimensionality Reduction for Neural Populations and Network Models”. In: *PLOS Computational Biology* 12.12, pp. 1–27. DOI: 10.1371/journal.pcbi.1005141.