Ideomotor feedback control in a recurrent neural network

Mathieu Galtier

February 17, 2014

Abstract

The architecture of a neural network controlling an unknown environment is presented. It is based on a randomly connected recurrent neural network from which both perception and action are simultaneously read and fed back. There are two concurrent learning rules implementing a sort of ideomotor control: (i) perception is learned along the principle that the network should predict reliably its incoming stimuli; (ii) action is learned along the principle that the prediction of the network should match a target time series. The coherent behavior of the neural network in its environment is a consequence of the interaction between the two principles. Numerical simulations show a promising performance of the approach, which can be turned into a local, and better "biologically plausible", algorithm.

1 Introduction

Animal life is characterized by the emergence of robust control mechanisms used in a large variety of environments. Beyond evolution which selects the life forms which make the most of their environment, it seems that some animals have the ability to quickly learn to interact constructively with new environments. Most probably, it means that the nervous systems of such animals can perform a sort of blind control of their environments: control is achieved without previous knowledge of the environment. It would surely be of great help to uncover such a mechanism, not only from a biological viewpoint, but also to replicate it for engineering tasks.

Control theory has been a vivid field of research for decades which has provided many useful applications. However, although linear systems are well understood [Kwakernaak and Sivan, 1972, Fortmann and Hitz, 1977], non-linear systems still require a significant effort to be dealt with [Slotine et al., 1991]. In particular, the control system is often assumed to have some explicit knowledge about the system to be controlled. When nothing is known about the system to be controlled, the most widespread method is to use a Proportional-Integral-Derivative controller. Indeed, computing the best inverse model [Jordan, 1996] from an unknown environment is a very difficult task which has received no universal answer so far.

Neural networks are bio-inspired mathematical object which have good learning capacities [Bishop, 1995]. A large number of control algorithms have used their adaptability to model some aspect of the environment and / or to design adaptive controllers. The challenge is to be able to internally predict the outcomes of a potential movement so that the best motor commands can be chosen [Kawato et al., 1987, Ge et al., 2008, Yang et al., 2008]. Typically, neural networks are used for two purposes: identification of the environment and the actual control which can be computed once a good estimate of the environment has been designed [Narendra and Parthasarathy, 1990].

A main drawback of feedforward neural networks is their inability to take time dependencies into account. Although the usual work-around is to use tapped delay lines, a more natural approach would be to use recurrent neural networks (RNN) which are more suited to dynamical systems approximation. Based on the two steps of identification and control, various RNN have been proposed as controllers e.g. [Chow and Fang, 1998, Wang and Hill, 2006, Prokhorov, 2007]. However, RNN learning is notoriously known to be slow and to be subject to problematic bifurcations [Doya, 1993, Pearlmuter, 1995].

*NeuroMathComp, Inria Sophia / UNIC, CNRS / Minds, Jacobs University Bremen. mathieu.galtier@inria.fr
To circumvent this problem recent algorithms [Pan and Wang, 2012, Waegeman et al., 2012] have been based on a reservoir computing architecture and more precisely Echo State Networks (ESN). They are recurrent networks where only the read-out form a random reservoir of neurons is learned [Jaeger, 2001, Jaeger and Haas, 2004]. In these networks, the slow convergence and the bifurcation issues are bypassed, but the mathematical understanding of the functioning of ESN is lacking. Nonetheless, ESN have proved to be very good at handling time dependencies and at predicting time series. Therefore, they provide a solid basis upon which this paper will design a novel control architecture.

Most neural networks for control used so far have been based on two networks: one for the estimation of the state of the environment (which can be called perception), and another for the design of the control (which can be called action). In this paper, I will introduce a somewhat different architecture, since it will only be made of a single recurrent neural network from which two read-outs are drawn and fed back. This is actually a fundamental feature of the approach, which resonates with the field of psychology called ideomotor theory [Greenwald, 1970, Shin et al., 2010]. This theory argues that perception and action are tightly linked and even represented in a single “domain”. More precisely a fundamental concept is that actions do not aim at changing the world directly, they rather aim at changing the perception of the world. Thus action and perception are deeply entangled and one could say that when the neural network “thinks” or predicts a future for its stimuli, then the corresponding action follows [Friston et al., 2010]. Alternatively, the conceptual difference from traditional feedback loop algorithms can be seen in the structure of the controller in figure 1: at the level of the controller information flows in both directions, what is usually called the output of the controller is also fed-back to the central network.

In this paper, I introduce an ideomotor recurrent neural network (IDRNN) together with a learning procedure in order to control an unknown environment. This paper intends to be a proof of concept that such a neural network can successfully learn to control fairly complicated dynamical systems. In section 2, I introduce the network and the notations. Section 3 will be devoted to explaining the computational principles underlying ideomotor learning and section 4 describes the corresponding algorithm. Numerical experiments for various environments and a short comparison with the ESN based method in [Waegeman et al., 2012] will be presented in section 5. Finally, I will discuss the biological plausibility of this method in section 6.

## 2 Model

The model details the dynamics of a recurrent neural networks interacting with an unknown environment. The way the environment interacts with the agent (through sensors and actuators) is also formally unknown. It is the role of the agent to understand this interaction through statistical observations.

### Notations and conventions:

Several elements of the neural network and the environment are time-dependent multidimensional functions which will be written with lower-case bold letters, e.g. \( \mathbf{v} \). The value of these functions at time \( t \) will be written as an index to the function's notation, e.g. \( \mathbf{v}_t \). Most of these functions will be described by recurrence equations which are assumed to start from \( t = 0 \). For simplicity, it is assumed that the value of all the functions introduced in the following are null on \( \mathbb{R}^- \). The matrices are written in upper case bold, e.g. \( \mathbf{W} \).

We now detail the different parts of the neural network and the environment:

- **Environment:**
  The environment possibly has a complicated non-linear and/or stochastic dynamics which we know nothing about. From the controller perspective, only some measures of the environment state are known. They are sometimes called observations but we refer to them as *stimuli* not to confuse them with the states of the reservoir which could also be called observations in the framework of online least mean square problems.

  The \( n_p \) neurons of the perceptive area receive information from the environment through the input vector \( \mathbf{u}_t \in \mathbb{R}^{n_p} \) at time \( t \in \mathbb{R}_+ \) (where \( n_p \) is the dimension of the perceptive area).
Figure 1: General structure of the proposed controller interacting with its environment. Learning only modifies the connections in red (from reservoir to perceptive and motor area).

- **Network activity:**
  We assume the controller is made of a neural network, which is decomposed in three parts: a *perceptive area*, a *reservoir* and a *motor area* see, figure 1. The variables describing the activity of these subsets of the neural network are respectively \( v^p_t \in \mathbb{R}^{n_p} \) called *perception*, \( v^r_t \in \mathbb{R}^{n_r} \) and \( v^a_t \in \mathbb{R}^{n_a} \) called *action*, where \( n_p, n_r, n_a \in \mathbb{N}^* \) are the number of neurons in the respective area. Note that the letters \( p, r, a \) will always stand for perceptive area, reservoir and motor area respectively. We also define the variable gathering the entire network activity:
  \[
  v = (v^p \ v^r \ v^a)^T \in \mathbb{R}^{n_p + n_r + n_a}.
  \]

  The full connectivity matrix of the network is
  \[
  W = \begin{pmatrix}
  0 & W^{pr} & 0 \\
  W^{rp} & W^{rr} & W^{ra} \\
  0 & W^{ar} & 0
  \end{pmatrix}
  \]

  which means that there are no connections between and within the perceptive and motor areas. We also define \( W^r = (W^{rp} \ W^{rr} \ W^{ra}) \in \mathbb{R}^{n_r \times (n_p + n_r + n_a)} \) corresponding to all the connections to the reservoir.

  A main idea in Echo State Networks [Jaeger, 2001] from which this work is inspired, it that the connections within and to the reservoir are randomly drawn. This means that the components of \( W^{rr}, W^{rp} \) and \( W^{ra} \) are i.i.d. constants along \( \mathcal{N}(0, \sigma^2), \mathcal{N}(0, \kappa^2) \) and \( \mathcal{N}(0, \gamma^2) \). Albeit surprising, this choice leads to relevant prediction results and cheap algorithms.

  For simplicity, perception and action are considered to be linearly related to the reservoir activity, which follows a classical neural network equation with linear external forcing from the perceptive and motor areas. Besides, the perceptive area is stimulated by a weighted mean between the stimuli and the current prediction of the network. With the notations introduced before, this reads
  \[
  \begin{align*}
  v^r_t &= (1 - lT)v^r_{t-1} + \tau W^r \begin{pmatrix}s(v^r_{t-1}) \\ v^p_{t-1} \\ v^a_{t-1}\end{pmatrix} \\
  v^p_t &= (1 - \alpha)W^{pr}v^r_t + \alpha u_t \\
  v^a_t &= W^{ar}v^r_t
  \end{align*}
  \]

  where \( s \) is an element-wise sigmoidal function, \( l \in \mathbb{R}_+ \) is a decay constant and \( \alpha \in [0, 1] \) balances the contribution two terms: the stimuli \( u \) and the term \( W^{pr}v^r_t \) which we call *prediction*. Note that if prediction and stimuli are identical then the perception \( v^p \) is also identical.
• **Target trajectory:**
The role of the neural network will be to control the environment so that it reproduces a target trajectory which we write $z_t \in \mathbb{R}^q$, where $q \leq n_p$. The target corresponds to (at least) one neuron in the perceptive area: if learning is successful, the stimuli to this subset of the perceptive neurons will be driven along the target trajectory. In the present formalism, the neural network can only control its stimuli (as opposed to an unobserved environment state).

• **Learning** corresponds to tuning the connections in the network. We assume that not all the connections are learnable: only the connections in red in figure 1 are learnable. In other words, and in a reservoir computing spirit, the connections to and within the reservoir $W^r$ are fixed. We call **perceptive learning** the tuning of the connectivity $W^{pr}$ and **motor learning** the tuning of the connectivity $W^{ar}$.

### 3 Principle of Ideomotor learning

In this paper, ideomotor learning is defined by the combination of two principles: (i) perceptive learning aims at minimizing the distance between internal predictions $W^{pr}, v^r$ and stimuli $u$, and, (ii) motor learning aims at minimizing the distance between internal predictions $W^{pr}, v^r$ and target $z$.

A control task is said to be successful when stimuli and target are equal. This can be a difficult task to achieve, in particular when the environment is unknown. The idea behind ideomotor learning is that having a good predictor of the stimuli may help designing an intelligent control. A controller which would be able to faithfully reproduce the stimuli, without needing to “see” them, would also know how to perturb the environment to reach a desired target. Therefore, one needs to learn a model of the world (principle (i) above) and, at the same time, make sure this model is going in the desired direction (principle (ii) above). More formally, ideomotor learning can be seen as adding a third variable, the prediction, in the distance between stimuli and target and using the triangular inequality to break the minimization of this distance into two sub-tasks.

The fact that motor learning makes no reference to the stimuli $u$ has important functional consequences. Indeed, the actions exclusively aims at modifying the reservoir dynamics so that the perception matches the target. The impact on the actions on the world and the stimuli $u$ is simply a byproduct of the method. In fact, the actions only want to control the internal model of the world that perceptive learning tries to build.

Mathematically, it is possible to formalize the two principles of ideomotor learning as

\begin{align}
\text{(i) perceptive learning} & \quad \text{minimize} \quad \sum_{s=0}^{t} \| u_s - W^{pr}, v^r_s \|^2 \\
\text{(ii) motor learning} & \quad \text{minimize} \quad \sum_{s=0}^{t} \| z_s - W^{pr}, v^r_s \|^2 \tag{2}
\end{align}

where $(u, v^r)$ is a solution of system (1) defined for given $W^{pr}$ and $W^{ar}$. The matrix $W^{p} \in \mathbb{R}^{q \times n_p}$ is the restriction of the perceptive matrix $W^{pr}$ to the dimensions which are to be controlled along the trajectory $z$, (i.e. to create $W^p$, some rows of $W^{pr}$ have been removed). If learning is perfect, i.e. both sums in (2) are 0 for all $t$ such that $(v^r, u)$ is on a limit cycle, then it is clear that the task is reached: $u_t = z_t$. We restrict our analysis to the case where such a limit cycle exists (which typically excludes non-controllable, non-observable environments).

Designing an online algorithm reaching this limit cycle reveals a fundamental problem: at time step $t$ the network has to figure out a new matrix $W^{pr}$ based on the potential impact it would have had in the past. But it cannot really know what this impact would have been since it would need to re-simulate the past with this new matrix. Another way to see this is to observe that the values of $u_t$ and $v^r_t$ depend on $W^{pr}$, and, therefore, perceptive learning is not a simple least square problem, where observations $v^r_t$ and target $u_t$ are not determined by the chosen weight vector. Actually, an exact online minimization procedure is out of reach simply because the neural network knows (a priori) nothing about the world and cannot simulate it.
4 Greedy RLS algorithm

In this paper, a greedy Recursive Least Square (RLS) approach to solve the least square problems is considered. It corresponds to dynamically updating the connections based on a RLS algorithm performing the online minimization of the following problem:

$$\text{minimize } H^p_{t} = \sum_{s=0}^{t} \lambda_{p}^{t-s}\|u_s - W^{pr}.v^p_s\|^2 + \mu_p\|W^{pr}\|^2$$

$$\text{minimize } H^a_{t} = \sum_{s=0}^{t} \lambda_{a}^{t-s}\|(W^p)^\dagger.(z_s - W^p.v^p_s)\|^2 + \mu_a\|W^{ar}\|^2$$

(3)

where $\lambda_p, \lambda_a \in [0,1]$ are forgetting factors and $u$ and $v^p$ are the solution of system (1) with time dependent connection matrices $W^{pr}_t$ and $W^{ar}_t$. If $\lambda$ is close to 0, then the sum simply involves very recent observations. The additional terms involving the squared norm of $W^{pr}$ and $W^{ar}$ correspond to the usual Tikhonov regularization and the numbers $\mu_p, \mu_a > 0$ control the amount of regularization. The notation $(W^p_W)^\dagger$ is the pseudoinverse of $W^pW$. The slight modification of the motor criterion in (2) due to the inversion of $W^pW$ simply corresponds to taking a different norm for the minimization. It is necessary to turn motor learning into a classical weighted least square problem. Indeed, one can unravel the dynamics of the reservoir for one time step to let the connection explicitly appear in the criterion. This corresponds to injecting the first row of (1) into $z_t - W^p.v^p_t$ which leads to

$$z_t - W^p.v^p_t = \tau(y_t - W^p.W^{ra}.W^{ar}.v^r_{t-1})$$

(4)

where $y_t = \frac{z_t - W^p.(1-\tau)v^r_{t-1} - W^p.W^{ra}.s(v^r_{t-1}) - W^p.W^{ra}.v^r_{t-1}}{\tau}$. As a consequence, $z_t - W^p.v^p_t$ appears as a linear function of $W^{ar}$ which will make it possible to use classical algorithms for minimization. Note that the particular dynamics of the reservoir in (1) was chosen such that such a linear problem would appear. It also explains why we cannot unravel the dynamics for more time steps: the reformulation into a least square problem would be impossible. From this formulation, it becomes clear that pre-multiplying the factor $z_t - W^p.v^p_t$ by $(W^p.W^{ra})^\dagger$ for the motor learning with the RLS algorithm in equation (3) is useful. Indeed, motor learning becomes the minimization of

$$H^a_t = \sum_{s=0}^{t} \lambda_{a}^{t-s}\|(W^p)^\dagger.y_s - W^{ar}.v^r_{s-1}\|^2 + \mu_a\|W^{ar}\|^2$$

(5)

which takes the form of a classic weighted least square problem which can directly be solved by a RLS algorithm. Note that matrix $W^p.W^{ra}$ has the size of the number of controlled dimensions (spatial dimension of the target) times the number of dimension for action, which is likely to be small enough to enable a computationally cheap inversion at each time step.

The greedy RLS algorithm consists in implementing (3) with a RLS algorithm which quickly forgets the past. Indeed, to circumvent the dependency of $u$ and $W^{pr}.v^p$ on $W^{ar}$ the choice of the forgetting factors $\lambda_p, \lambda_a$ is crucial. When the network starts from an uninformed state (e.g. the null state), it is important for them to have small values, typically 0.99. A rule of thumb [Haykin, 2005] to tune them is that the memory of the algorithm roughly corresponds to $\frac{1}{1-\lambda}$ time steps.

The regularized RLS algo is recalled in this paragraph. It is an algorithm which recursively solves the following generic problem

$$\text{minimize } H_t = \sum_{s=0}^{t} \lambda^{t-s}\|x_s - W_t.y_s\|^2 + \mu\|W\|^2$$

(6)
It can be solved by iterating the following regularized RLS step [Haykin, 2005, Gunnarsson, 1996] as new targets \( x \) and new observation \( y \) arrive. This algorithm has already been successfully used for Echo State Networks [Jaeger and Haas, 2004, Sussillo and Abbott, 2009, Laje and Buonomano, 2013]. The algorithm corresponding to one step of the RLS algorithm is given in algorithm 1. In this algorithm,

\[
P \in \mathbb{R}^{n \times n} \text{ is the inverse correlation matrix of the observations, which correspond to the reservoir activity in this paper.}
\]

Pseudo-code Therefore, the algorithm summarizing the update of the greedy RLS neural network is described in algorithm 2.

Algorithm 2 IDRNN

5 Numerical experiments

In this section, I show on different examples that the proposed architecture can effectively control an unknown environment along a desired trajectory. It is important to realize that the following numerical simulations correspond to a neural network which learns to control the environment: at time \( t = 0 \) the neural network has no clue about what system it is dealing with. Thus the results are not to be compared to a classical control setup where the controller was previously tuned to the specific environment.

5.1 Learning to control random neural networks

As toy models of the environment, I first choose randomly connected neural networks from which a linear read-out is to be controlled along a sinus function. It has been argued that this class of systems is very rich since it can approximate any dynamical system [Sontag, 1997]. I do not claim that the
IDRNN can control any such random neural network since they can become very complicated, but it performs well on reasonably complicated instances of such networks. I will consider neural networks made of 5 neurons from which a one-dimensional linear readout is computed. The equation governing by these driven random neural networks is:

\[
\begin{align*}
    x_{t+1} &= (1 - \bar{l}\bar{\tau})x_t + \tau \bar{W} \cdot \tanh(x_t) + b \cdot v_t \\
    u_t &= c' \cdot x_t
\end{align*}
\]

where matrix \( \bar{W} \in \mathbb{R}^{5 \times 5} \) and the vectors \( b, c \in \mathbb{R}^5 \) have components which are drawn i.i.d. \( \mathcal{N}(0, \bar{\sigma}^2) \), \( \mathcal{N}(0, \bar{\kappa}^2) \) and \( \mathcal{N}(0, \bar{\gamma}^2) \) respectively.

With different choices of parameters, one can get very different resulting dynamics as shown in figures 2(a), 3(a), 4(a). Even with identical parameters the different realizations of the connections can lead to qualitatively different dynamics. I chose three cases corresponding to converging (figure 2), oscillatory (figure 3) and diverging (figure 4) situations.

For all the simulations, I chose the same parameters for the IDRNN: \( n_a = n_p = 1, n_r = 100, \sigma = 1.5, \kappa = \gamma = 0.1, l = 1, \tau = 0.1, \alpha = 0.5, \eta = 1, s(x) = \tanh(x + 0.01), \lambda_p = 0.99, \lambda_a = 0.9, \) \( \mu_p = 10^{-6}, \mu_a = 10^{-3} \). The parameters of the environment are reported in the figures.
In each situation, the very same neural network has managed to control qualitatively different environments it knew nothing about along the same trajectory $z_t = \sin(2\pi t/100)$. The $L_1$ error between stimuli and target over the last 100 time steps (corresponding to a period of the target) is respectively 0.0006, 0.0046 and 0.0017 for converging, oscillatory and diverging situations, meaning that the control is successful.

In each case, the first few hundred time steps display an extremely variable behavior. The network has difficulties choosing relevant actions since its perception is not fully developed yet. Although the variations look completely unstructured the network still manages to calm down and converges to the desired trajectory. Unfortunately, this is not always the case: the search period can bring the environment to an undesired location in the state space leading to a failure of the control. For instance, controlling a quickly diverging environment (not shown) can fail when the environment diverges faster than the network can learn. The two important parameters to control the behavior of the network in the beginning are the initial values of $W^{pr}$ and $P_p, P_a$. If the value of $W^{pr}_0$ is too small then the pseudo-inversion of $W^{pr} W^{ra}$ in the algorithm can lead to extremely large control values in the very beginning of the simulation. Similarly, choosing large initial values for $P_p$ or $P_a$ induces a large variability at the beginning of the simulation [Haykin, 2005] which can lead to poor results.

Observe in figure 4(b) that the hidden variables in the environment $x$ are not displayed. Indeed, although the network is controlling properly the environment read-out $u$ it does not necessarily prevent these hidden variables from diverging. This leads to a numerical overflow in the simulation after a certain time and an increased variability due to numerical round-offs. To prevent this one would need to ask the IDRNN to control also the states $x$.

### 5.2 Learning to control delayed systems

Actually, the IDRNN in its current form fails at controlling delayed systems or even some instantaneous systems which for which the impact of an action may take some time steps to reveal itself. This is not a surprise since the network only tries to control what it happening from one time step to the other.

A simple extension of the algorithm 2 makes the IDRNN able of controlling such delayed systems. The idea is to learn to predict the future of the stimuli at $t + \delta$ time steps with $\delta > 1$. This corresponds to adding a third readout matrix $W^{fr}$ to the network in figure 1. The predicted value $W^{fr} v_t^r$ should be a approximation of $u_{t+\delta}$ and is not fed back to the network. In practice, learning such a future prediction matrix can be done by applying an RLS algorithm to approximate $u_t$ based on the observation $v_{t-\delta}^r$. This implies to store the history of the reservoir during $\delta$ time steps. Once this prediction is set-up it suffices to replace $W^p$ by $W^{fr}$ in algorithm 2. This leads to a delayed version of the IDRNN detailed in algorithm 3 which does have a time window similar to [Waegeman et al., 2012].

With this new algorithm, it is possible to control systems for which the instantaneous IDRNN
failed. Here I consider two examples of such systems: first a linear oscillatory system with a single control variable and, second, a random neural network with delayed action.

In a first time, we consider the following environment to control

\[
\begin{pmatrix}
u_{t+1} \\ x_{t+1}
\end{pmatrix} = \begin{pmatrix} u_t \\ x_t 
\end{pmatrix} + 0.1 \begin{pmatrix} -0.05 & -1 \\ 1 & -0.05 \end{pmatrix} \begin{pmatrix} u_t \\ x_t \end{pmatrix} + 0.1 v^a_t \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}
\]

This corresponds to a linear oscillatory system as shown in figure 5(a) somewhat similar to the first system studied in [Waegeman et al., 2012]. The negative terms on the diagonal compensate for the tendency of the Euler algorithm to diverge when simulating linear systems and can be disregarded for the interpretation. Actually, the difficulty to control the system comes from the control vector (1 0.5) which is positive for both components. To understand intuitively the problem it is useful to see that one dimension is excitatory while the second is inhibitory. When the control variable only makes it possible to positively stimulate both dimensions, it is not clear what will be the situation in a more complex scenario.

In the end, one needs to see a bit more in the future to understand the impact of the action. This is precisely why adding the time window \( \delta \) to the prediction makes the task much more simple. Indeed, the IDRNN can effectively control the system as shown in figure 5(b). The parameters used are the same as previously with the addition of \( \delta = 20, \lambda = 0.995 \) and \( \eta = 10^{-4} \).

The second environment is governed by a random neural network as previously but with a delayed command:

\[
\begin{align*}
x_{t+1} &= (1 - l_T)x_t + \tau W. \tanh(x_t) + b.v^a_{t-50} \\
u_t &= c'.x_t
\end{align*}
\]

The parameters of the IDRNN are identical to before except that \( \delta = 55, \lambda = 0.995 \) and \( \eta = 10^{-4} \) which explains the great variability at the beginning of the simulation.

The control performed by the IDRNN is displayed in figure 6. Although the control takes more time (4000 time steps instead of 1500), it finally manages to bring the system along the desired trajectory.

5.3 Short comparison with the double reservoir architecture

Although an extensive comparison between the IDRNN and the double reservoir architecture (DRA) [Waegeman et al., 2012] is beyond the scope of this paper, I show here that they perform similarly on
Figure 5: Learning to control a linear oscillatory system with a single positive control variable. See figure 2 for the color code.

Figure 6: Learning to control of a random neural network with a delayed control variable. See figure 2 for the color code. Environment parameters: $\bar{\sigma} = \bar{\kappa} = \bar{\gamma} = 1$, $\bar{l} = 1$, $\bar{\tau} = 0.1$. 

the task of controlling a heating tank.

The system to control, a heating tank, is the same as the second example in [Waegeman et al., 2012] and the interested reader should refer to this paper for the implementation details (I took the same parameters). Briefly, the system is a non-linear state-delayed system where the delay depends on the action. It corresponds to controlling a heating tank whose input is a stream of cold water. The water is heated in the tank and is then channeled through a tube to a destination where the output temperature is measured. When the water throughput (corresponding to the action $v^a$) is increased the output temperature decreases, but it takes less time to get out of the tube. It is shown in [Waegeman et al., 2012], that the DRA outperforms another algorithm, called NEPSAC [Gálvez-Carrillo et al., 2009], on this task.

In this paper, there are two differences from the heating tank used in [Waegeman et al., 2012]: first, the stimuli to the network are scaled to meet the networks dynamics. I take as input $\frac{T-30}{5}$ where $T$ is the output of the heating tank simulation. Second, the target trajectory is different, see figure 7(a). It is much faster so that the results of the control algorithm do not look as good as in the original paper presenting the DRA. I have not tried to tune the parameters to get the best results, since I only intended to show that both algorithm would perform similarly on this task with naive parameter tuning.

Actually, the IDRNN failed to control the system when it is randomly initialized. Interestingly the DRA could control the system although I did not use any babbling initialization as discussed in [Waegeman et al., 2012]. To get good results with the IDRNN I had to properly initialize the network. To do so I forced the network to reproduce the actions of the DRA during 3000 time steps. This was done using a simple ESN architecture. At the end of initialization the system had a behavior which was very different from the DRA, see figure 7(b). Nonetheless it was sufficient for the coupled system to be in an attractor and the control did work.

![Figure 7](image-url) (a) Target trajectory $z$. (b) Initialization of IDRNN.

Figure 7: (left) target trajectory used for the heating tank experiment. (right) Result of the initialization of the IDRNN. The actual implementation of algorithm 3 started right after. In blue are the inputs $u_t$ and in pink are the future inputs $u_{t+\delta}$. In grey are the actions $v^a$. In green is the target for the initialization phase (different from $z_d$). Note that it is a target for the action and not the perception. In red is the perception $v^p$ and in cyan in the prediction of the future $v^f$.

I simulated the IDRNN and the DRA control for 50000 time steps and obtained a performance displayed in figure 8. It appears that the DRA converges faster to a good control behavior, see 9(b) but its performance slowly deteriorates with time. On the contrary, the IDRNN converged more slowly but its performance improves with time. Eventually, the system behavior (see 9(a)) is similar to the best case for the DRA, although a bit worse. The parameters used are $\delta = 20$, $n_r = 100$, $\sigma = 1.5$, $\kappa = \gamma = 0.1$, $l = 1$, $\tau = 0.1$, $\alpha = 0.5$, $\eta = 1$, $s(x) = \tanh(x+0.01)$, $\lambda_p = \lambda_a = \lambda_f = 0.9999$, $\mu_p = \mu_f = \mu_a = 10^{-6}$. The reason why I take the forgetting parameters $\lambda_p = \lambda_a = \lambda_f$ to be so large is because an initializing method is used and I want the network not to forget immediately the initialization. I also reset the matrix $P_a$ to $0.002I_d$ at the end of the initialization to slow learning.
down to stay close to initialization at first. The parameters for the DRA are identical to that of the IDRNN.

![Figure 8: Control performance of IDRNN (red) and DRA (blue). The $L_1$ distance between the target $t \mapsto z_t$ and $t \mapsto u_{t+\delta}$ over a siding window of size 900 is used for this performance index. The two dots correspond to figure 9.](image)

To summarize, the two methods seem to roughly give similar results at their optimum. However, they seem to differ in the evolution of the performance with time.

6 Biological plausibility

For a theoretical work, it is difficult and dangerous to claim biological plausibility, especially because brains are significantly more complicated than the simple networks usually considered in the literature. There are obvious situations where a particular algorithm is not biologically plausible such as the DRA which requires to have two different reservoirs with the same internal and output weights at all
times. When such situations are ruled out biological plausibility is often reduced to requiring that computation is local. Concretely, learning the connection between two neurons should be independent of any other neuron. Computationally, it is a strong requirement, which rejects several widespread algorithms. In particular, the IDRNN as defined by algorithms 2 and 3 are clearly non-local since they involve the computation of the inverse of the correlation matrix of the reservoir.

However, the ideomotor principle might be implemented in the brain by a greedy Least Mean Square (LMS) algorithm instead of a RLS algorithm. LMS corresponds to a simple stochastic gradient descent of the principles in (2). It leads to slower convergence times than the RLS algorithm [Haykin, 2005], but both aim at solving the same problem.

The ideomotor LMS algorithm can be directly derived from (2) under the greedy assumption that \( u \) and \( v^r \) do not depend on \( W^pr \) and \( W^ar \). It comes as the gradient descent of \( H^p \) and \( H^a \) and can be written as

\[
W^pr_{t+1} = W^pr_t + \epsilon_p(u_t - W^pr_t v^p_t) v^p_t'
\]

\[
W^ar_{t+1} = W^ar_t + \epsilon_a(W^p_t W^ra_t)'(y_t - W^p_t W^ra_t W^ar_t v^r_{t-1}) v^r_{t-1}'
\]

where \( \epsilon_p \) and \( \epsilon_a \) have been re-parametrized to absorb constants.

Before studying this locality constraint, the biological relevance of the existence of a target time series has to be discussed. Indeed, it is biologically dubious to assume that the brain sets itself target time series to track. However, the formalism can be slightly reinterpreted so as to describe a more realistic situation: we could simply ask the network to maximize one of its stimuli which would therefore be seen as a reward neuron. This would correspond to replacing \( z_t \) by 0 and taking the opposite of the second term in the last row of equation (7) leading to \( W^ar_{t+1} = W^ar_t + \epsilon_a(W^p_t W^ra_t)'(z_t - W^p_t v^p_t) v^p_t' \).

In this framework, one could see the network as solving a reinforcement task trying to maximize its reward (which is assumed to be precomputed at each time step and given as a stimulus).

It is easy to see that perceptive learning is local. Indeed, the modification of the reward neuron and one motor neuron as in figure 10. The extension to the multi-dimensional case is straightforward. In this case, both \( W^p_t W^ra_t \) and \( W^p_t v^p_t \) are real numbers which need to be communicated to the motor connections. Therefore, we assume that there is a modulatory connection going from the reward neuron to the motor connections which provides them with the value of \( W^p_t v^p_t \). Thus, it only remains to be shown how \( W^p_t W^ra_t \) can be computed. I propose an online approximation of this quantity, which we write \( b_t \equiv W^p_t W^ra_t \) as the solution of \( b_t = \arg \min_{c \in \mathbb{R}} \sum_{s=0}^{t} \beta^{t-s} \| W^p_s v^r_s - c v^a_{s-1} \|^2 \)

where \( \beta \in [0, 1] \). This approximation can be estimated in an online framework through another LMS algorithm:

\[
b_{t+1} = b_t + \epsilon_s(W^p_t v^p_t - b_t v^a_{t-1}) v^a_{t-1}
\]

And therefore, the LMS update of the motor connections in system (7) can be written as a local learning rule as

\[
W^ar_{t+1} = W^ar_t + \epsilon_a b_t v^p_t v^p_t'
\]

The combination of the last two equations form a local update of the motor connections in the case of a single reward neuron and a single motor neuron.

Interestingly, LMS and RLS can be shown to be equivalent if the reservoir has a temporal correlation matrix proportional to the identity [Farhang-Boroujeny, 1998]. This regime has been often observed in biology, where in vivo cortical tissues are said to be in an asynchronous irregular state [Ecker et al., 2010, Renart et al., 2010].
Figure 10: Diagram showing the necessary wiring of the network if learning has to be local, in the illustrative case of a single reward neuron and a single motor neuron. The perceptive area is made of a reward neuron and other perceptive neurons, all of which receive stimuli from the environment (possibly some other part of the brain). The reward neuron has to send a modulatory synapse to the motor connections. The information from the motor neuron is only used for learning and does not modify directly the dynamics of the motor neuron.

7 Conclusion

This paper defines a recurrent neural network which blindly learns to control an unknown environment. Based on a randomly connected reservoir it learns on the fly two read-outs which correspond to perception and action. These read-outs are learned according to two principles: perceptive learning corresponds to maintaining good predictions of the incoming stimuli; motor learning tries to change the dynamics of the reservoir so that the stimuli predictions match a target trajectory. Actually, the control of the environment is just a byproduct of the behavior of the neural network which only cares about its own predictions. This algorithm is closely related to the ideomotor theory and provides a computational implementation of these concepts.

Several numerical simulations have established a proof of concept for this neural network. It manages to control fairly complicated dynamical systems and properly handles non-linearities and delays. Importantly, it falls in the small class of control schemes which do not need knowledge about the system they are controlling. It has been shown on an example that the performances of this architecture were similar to that of the DRA algorithm. Although the IDRNN needs to be initialized more carefully than the DRA, it shows more stability once in the attraction basin. However, the rigorous benchmarking of both methods with other approaches should be carried out in future research.

An important characteristic of these two reservoir based algorithms which has not been discussed so far is their ability to embed existing control schemes. Indeed, traditional ESN algorithms can be used for these architectures to reproduce any trajectories of both perception and action. Indeed, provided with the stimuli and actions of any control scheme, it is possible to build a neural network either IDRNN or DRA which reproduces this controller (only on the observed trajectories of course). This could serve as an interesting initialization procedure for these network which can therefore improve a lot of existing control algorithms.

The IDRNN can be said to be closer to biology than the DRA, since there is no sharing of weights between two networks, and one might see some biological plausibility to its LMS alternative. It is intriguing that it requests a certain network architecture which shares some similarity with some organs in biology and notably the spinal chord [Butler and Hodos, 2005]. Indeed, in both networks there is one central recurrent network and two types of read-out: the dorsal root for perception and the ventral root for action. However, I think the link with biology is yet tenuous and should be
further investigated. An interesting perspective could be to add some biological structure (e.g. spiking neurons, micro-circuitry) in the network and see how it changes the performance.

The mathematical understanding of the properties of such a controller is a challenge which will have to be tackled in the future. The main difficulties so far are linked to the reservoir dynamics which is not well understood and to the greediness of the algorithm which necessarily implies some approximations which will have to be controlled.

This algorithm could be improved or extended in different ways. For instance, how a reinforcement learning approach (with distal rewards [Jordan and Rumelhart, 1992]) could be designed for a similar architecture? Another idea would be to build architectures from this building block: how could such networks interact and control each other into a coherent structure?

Aknowledgments: I thank Herbert Jaeger, Michael Thon, Jochen Steil, Felix Reinhart and Benjamin Schrauwen for helpful discussions. I was funded by the European project AMARSI.

References

[Bishop, 1995] Bishop, C. M. (1995). Neural networks for pattern recognition. Oxford university press.

[Butler and Hodos, 2005] Butler, A. B. and Hodos, W. (2005). Comparative vertebrate neuroanatomy: evolution and adaptation. John Wiley & Sons.

[Chow and Fang, 1998] Chow, T. W. and Fang, Y. (1998). A recurrent neural-network-based real-time learning control strategy applying to nonlinear systems with unknown dynamics. Industrial Electronics, IEEE Transactions on, 45(1):151–161.

[Doya, 1993] Doya, K. (1993). Bifurcations of recurrent neural networks in gradient descent learning. IEEE Transactions on neural networks, 1:75–80.

[Ecker et al., 2010] Ecker, A. S., Berens, P., Keliris, G. A., Bethge, M., Logothetis, N. K., and Tolias, A. S. (2010). Decorrelated neuronal firing in cortical microcircuits. Science, 327(5965):584–587.

[Farhang-Boroujeny, 1998] Farhang-Boroujeny, B. (1998). Adaptive filters: theory and applications. John Wiley & Sons, Inc.

[Fortmann and Hitz, 1977] Fortmann, T. E. and Hitz, K. L. (1977). An introduction to linear control systems. Crc Press.

[Friston et al., 2010] Friston, K. J., Daunizeau, J., Kilner, J., and Kiebel, S. J. (2010). Action and behavior: a free-energy formulation. Biological cybernetics, 102(3):227–260.

[Gálvez-Carrillo et al., 2009] Gálvez-Carrillo, M., De Keyser, R., and Ionescu, C. (2009). Nonlinear predictive control with dead-time compensator: Application to a solar power plant. Solar energy, 83(5):743–752.

[Ge et al., 2008] Ge, S. S., Yang, C., and Lee, T. H. (2008). Adaptive predictive control using neural network for a class of pure-feedback systems in discrete time. Neural Networks, IEEE Transactions on, 19(9):1599–1614.

[Greenwald, 1970] Greenwald, A. G. (1970). Sensory feedback mechanisms in performance control: with special reference to the ideo-motor mechanism. Psychological review, 77(2):73.

[Gunnarsson, 1996] Gunnarsson, S. (1996). Combining tracking and regularization in recursive least squares identification. In IEEE CONFERENCE ON DECISION AND CONTROL, volume 3, pages 2551–2552. Citeseer.

[Haykin, 2005] Haykin, S. S. (2005). Adaptive Filter Theory, 4/e. Pearson Education India.
[Jaeger, 2001] Jaeger, H. (2001). The “echo state” approach to analysing and training recurrent neural networks—with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report, 148:34.

[Jaeger and Haas, 2004] Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science, 304(5667):78–80.

[Jordan, 1996] Jordan, M. I. (1996). Computational aspects of motor control and motor learning. Handbook of perception and action: motor skills, 2:71–118.

[Jordan and Rumelhart, 1992] Jordan, M. I. and Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. Cognitive science, 16(3):307–354.

[Kawato et al., 1987] Kawato, M., Furukawa, K., and Suzuki, R. (1987). A hierarchical neural-network model for control and learning of voluntary movement. Biological cybernetics, 57(3):169–185.

[Kwakernaak and Sivan, 1972] Kwakernaak, H. and Sivan, R. (1972). Linear optimal control systems, volume 1. Wiley-Interscience New York.

[Laje and Buonomano, 2013] Laje, R. and Buonomano, D. V. (2013). Robust timing and motor patterns by taming chaos in recurrent neural networks. Nature neuroscience, 16(7):925–933.

[Narendra and Parthasarathy, 1990] Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. Neural Networks, IEEE Transactions on, 1(1):4–27.

[Pan and Wang, 2012] Pan, Y. and Wang, J. (2012). Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. Industrial Electronics, IEEE Transactions on, 59(8):3089–3101.

[Pearlmutter, 1995] Pearlmutter, B. A. (1995). Gradient calculations for dynamic recurrent neural networks: A survey. Neural Networks, IEEE Transactions on, 6(5):1212–1228.

[Prokhorov, 2007] Prokhorov, D. V. (2007). Training recurrent neurocontrollers for real-time applications. Neural Networks, IEEE Transactions on, 18(4):1003–1015.

[Renart et al., 2010] Renart, A., de la Rocha, J., Bartho, P., Hollender, L., Parga, N., Reyes, A., and Harris, K. D. (2010). The asynchronous state in cortical circuits. Science, 327(5965):587–590.

[Seamans and Durstewitz, 2008] Seamans, J. and Durstewitz, D. (2008). Dopamine modulation. Scholarpedia, 3(4):2711.

[Shin et al., 2010] Shin, Y. K., Proctor, R. W., and Capaldi, E. (2010). A review of contemporary ideomotor theory. Psychological bulletin, 136(6):943.

[Slotine et al., 1991] Slotine, J.-J. E., Li, W., et al. (1991). Applied nonlinear control, volume 199. Prentice-Hall Englewood Cliffs, NJ.

[Sontag, 1997] Sontag, E. (1997). Recurrent neural networks: Some systems-theoretic aspects. Dealing with complexity: A neural network approach, pages 1–12.

[Sussillo and Abbott, 2009] Sussillo, D. and Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. Neuron, 63(4):544–557.

[Waegeman et al., 2012] Waegeman, T., Wyffels, F., and Schrauwen, B. (2012). Feedback control by online learning an inverse model. Neural Networks and Learning Systems, IEEE Transactions on, 23(10):1637–1648.

[Wang and Hill, 2006] Wang, C. and Hill, D. J. (2006). Learning from neural control. Neural Networks, IEEE Transactions on, 17(1):130–146.
[Yang et al., 2008] Yang, C., Ge, S. S., Xiang, C., Chai, T., and Lee, T. H. (2008). Output feedback neural network control for two classes of discrete-time systems with unknown control directions in a unified approach. *Neural Networks, IEEE Transactions on*, 19(11):1873–1886.