Using conceptors to transfer between long-term and short-term Memory

Anthony Strock[2,1,3,*], Nicolas Rougier[1,2,3], and Xavier Hinaut[1,2,3]

[1] INRIA Bordeaux Sud-Ouest, Bordeaux, France
[2] LaBRI, Université de Bordeaux, CNRS, UMR 5800, Talence, France
[3] IMN, Université de Bordeaux, CNRS, UMR 5293, Bordeaux, France
* Corresponding author: Anthony.Strock@inria.fr

Abstract. We introduce a model of working memory combining short-term and long-term components. For the long-term component, we used Conceptors in order to store constant temporal patterns. For the short-term component, we used the Gated-Reservoir model: a reservoir trained to hold a triggered information from an input stream and maintain it in a readout unit. We combined both components in order to obtain a model in which information can go from long-term memory to short-term memory and vice-versa.

1 Introduction

Jaeger recently showed how to store and retrieve several temporal patterns using an extension of reservoirs [1, 2]. The key idea is to project, using Conceptors, the network activity into a lower dimensional space specific to the patterns. These Conceptors can be considered as a long-term memory (of the temporal patterns). In the meantime, we have shown [3] how a reservoir, using a gating signal, can faithfully memorize an information for a short delay (i.e. working memory) from a stream of random inputs. In this model, the maintenance of information in readout unit(s) is remarkably precise for rather long periods of time. However, for very long periods of time, the precision is lost because of a slow drift in the dynamics. In the present work, we introduce a preliminary attempt to link short-term and long-term memories by combining these two approaches: (1) a gated reservoir maintaining short-term information and (2) several conceptors maintaining long-term information.

2 Methods

We consider a reservoir of 1000 neurons that has been trained to solve a gating task described in [3]. In this task the model receives an input continuously varying over time (the value) and another input being either 0 or 1 (the trigger). To succeed the task, the output has to be updated to the value received when the trigger is active but has to stay still otherwise (similarly to a line attractor). In other words, the trigger acts as a gate that controls the entry of the value in the memory (the output). The overall dynamics of the network we consider is described by the following equations:

$$x[n] = C\tanh\left(W_{in}u[n] + W(x[n-1] + \xi) + W_{fb}(y[n-1])\right) \text{ and } y[n] = W_{out}x[n] \quad (1)$$

where $u[n]$, $x[n]$ and $y[n]$ are respectively the input, the reservoir and the output at time n. W, W_{in}, W_{fb}, W_{out} and C are respectively the recurrent, the input, the feedback, the output and the conceptor weight matrices and ξ is a uniform white noise term added to reservoir units. W, W_{in}, W_{fb} are uniformly sampled between -1 and 1. Only W is modified to have sparsity level equal to 0.5 and a spectral radius of 0.1. When W_{out} is computed to solve the gating task, the conceptor C is considered to be fixed and equal to the identity matrix ($C = I$). In normal mode, the conceptor C is equal to a conceptor C_m that is generated and associated to a constant value m. In order to compute this conceptor C_m, we impose a trigger ($T = 1$) as well as the input value ($V = m$) at the first time step, such that the reservoir has to maintain this value for 100 time steps. During these 100 time steps, we use the identity matrix in place of the conceptor. The conceptor C_m is then computed according to $C_m = XX^T\left(XX^T + \frac{I}{a}\right)^{-1}$, where X corresponds to the concatenation of all the 100 reservoir states after the trigger, each row corresponding to a time step, I the identity matrix and a the aperture. In all the experiments the aperture has been fixed to $a = 10$. For the conceptors pre-computed in Figure 1 and 2, the reservoir have been initialised with its last training state.

3 Results

Transfer between long-term and short-term memory: Figure 1 displays the two core ideas of our approach: **(1)** How to transfer short-term to long-term memory and **(2)** How to retrieve (in short-term memory) an information stored in long-term memory. **(1)** The long-term memory we consider is the conceptor C_m associated to the value m maintained in short-term memory. We compute C_m using the 100 first time steps after a trigger without conceptors ($C = I$), after what we update C with C_m. On figure 1B we can see that it doesn't seem to cause any interference in the short-term memory. However now the memory lies both in the conceptor C (long-term) and in the output unit y (short-term). A more extensive and quantitative analysis could study whether applying C_m would be a way to stabilize the short-term memory. **(2)** Now, the long-term memory we consider are only conceptors C_m^D associated to discrete values between -1 and 1 (11 values uniformly spread between -1 and 1). Similarly as before, after a trigger we compute a new conceptor C_m^D using the 100 first time steps and without conceptors ($C = I$). Then, we search for the closest conceptor C_m^i among the conceptors with discrete values C_m^D using a distance between conceptors and we update C with this conceptor. On figure 1C, we see the following behavior: after a trigger, the value is correctly updated in short-term memory and remains stable until C is updated (after 100 time steps) and then the output jumps to the closest discrete representation of the memory.

Generalizing long-term memory: On figure 2, we show two main ideas: **(1)** how a linear interpolation between two conceptors can allow to generalize to gating of other values, and **(2)** a representation of the space in which lies the conceptors and their link to the memory they encode. **(1)** Interpolation and extrapolation C of conceptor $C_{0.1}$ and conceptor $C_{1.0}$ has been computed as $C = \lambda C_{1.0} + (1-\lambda)C_{0.1}$ with 31

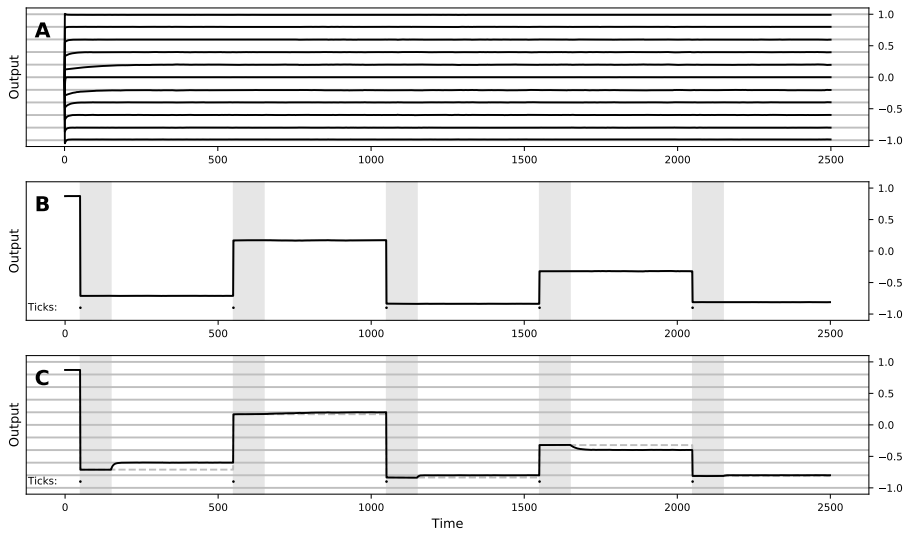

Fig. 1. Approximation with conceptors or discrete conceptors. Black: Evolution of the short-term memory (i.e. the readout y). Gray lines: the discrete value considered. Light gray areas: time when conceptors are computed for the current value stored in short-term memory. **A.** Discrete conceptors are appplied **B**-**C.** Exact conceptors are computed using 100 time steps after a trigger while C = I. **B.** The exact conceptor is applied to the following time steps. **C.** The closer conceptor among the discretized conceptor is applied for the following time steps. Dashed lines represents the memory that should have been kept if not discretized.

λ values uniformly spread between -1 and 2. Even though the interpolated ($\lambda \in [0,1]$) conceptors obtained are not exactly equivalent to C_m conceptors obtained in figure 1, they seem to also correspond to a retrieved long-term memory value to be maintained. The mapping between λ and the value is non-linearly encoded. For right-extrapolation ($\lambda \in [1,2]$) the conceptor seems to be linked to a noisy version of a C_m conceptor. A value seems still to be retrieved from long-term memory and maintained in short-term memory: the output activity is not constant, but its moving average is constant. For left-extrapolation ($\lambda \in [-1,0]$), the conceptor obtained seems not to encode any more information than the conceptor C_0: all the output activities collapse to zero. **(2)** Principal component analysis have been performed using 201 pre-computed conceptors associated to values uniformly spread between -1 and 1. The first three components explain already around 85% of the variance. The first component seems to non-linearly encode the absolute value of the memory (figure 2B) whereas the second component seems to non-linearly encode the memory itself (figure 2C). The C_m conceptors form a line but not a straight line (figure 2E-G), that might explain why extrapolation doesn't work as expected.

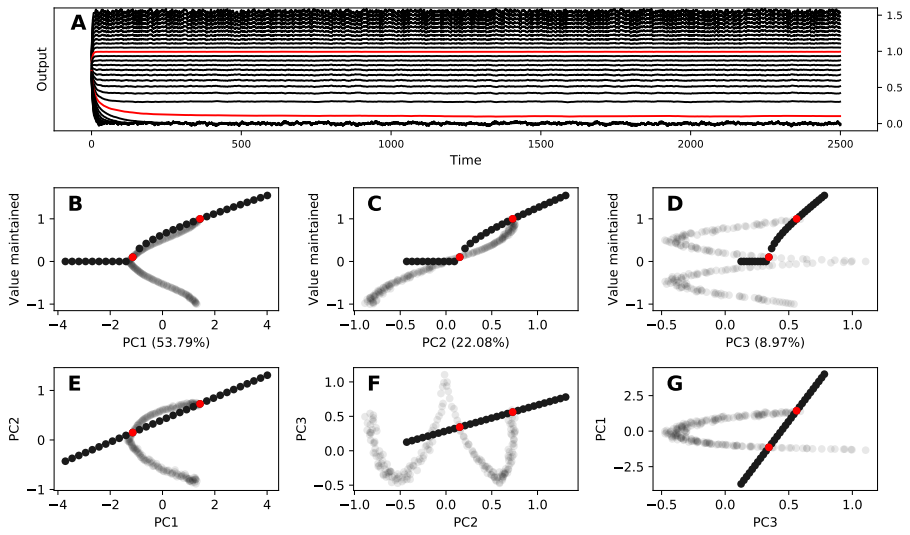

Fig. 2. Generalisation of C_m conceptors. Red: two learned conceptors $C_{0.1}$ and $C_{1.0}$. Black: Linear interpolation between conceptor $C_{0.1}$ and conceptor $C_{1.0}$. Gray: learned C_m for 201 value of m uniformly spread between -1 and 1. **A** Evolution of the short-term memory against time for the different conceptors. **B**-**D** Link between principal components of the constant conceptors and the memory it is encoding. For the interpolated conceptors, the memory is considered as the mean in the last 1000 time steps. **E**-**G** Representation of the conceptors in the three principal components of the C_m conceptors.

4 Discussion

This preliminary study introduces the basis for establishing a link between long-term and short-term memory. Future work will concentrate on removing the engineered steps, namely, the offline computation and selection of the closest conceptor. This could be realized using the autoconceptors introduced in [2]. Moreover, this study raised concerns concerning the best way to interpolate and extrapolate conceptors since, as we have shown, a mere linear combination might not represent the best solution.

References

1. Jaeger, H.: Controlling recurrent neural networks by conceptors. arXiv preprint arXiv:1403.3369 (2014)
2. Jaeger, H.: Using conceptors to manage neural long-term memories for temporal patterns. Journal of Machine Learning Research **18**(13), 1–43 (2017)
3. Strock, A., Hinaut, X., Rougier, N.P.: A robust model of gated working memory. biorXiv (2019). https://doi.org/10.1101/589564