# Decision making in low-rank recurrent networks

Nguyen Tien Dung Otten, Tobiasz Budzyński

July 30, 2021

## 1 The low-rank framework

Connecting behavior to neural data requires models. Those were often build according to the phenomenological low-dimensional dynamics hypothesis which was further applied in small and bigger models and compared to neural data.

A different approach to obtain a model was proposed by Omri Barak [3], by first training a recurrent neural network with low-rank connectivity matrix. Secondly, applying the dimensionality reduction technique - finding Gaussian parameters (multivariate Gaussian mixture models [2]) and proving the Gaussian nature of connections by checking the accuracy of the resampled model and reduced dynamics. The obtained hypothetical dynamics are intrinsic and aposteriori. With an exception of choosing which dynamics are interpretable, it is an automation of the scientific process [3].

### 1.1 The dynamics and dimensionality reduction

For a network with a recurrent layer of size $N$, the rank $R$ connectivity matrix is constructed by specifying $R$ pairs of patterns $\mathbf{m}^{(r)}$, $\mathbf{n}^{(r)}$ for $r = 1, ..., R$. An increment in dynamics takes the value of the scalar product of the network's state $\mathbf{x}$ and the vector $\mathbf{n}^{(r)}$, in the direction of the vector $\mathbf{m}^{(r)}$. The patterns (vectors) $\mathbf{n}^{(r)}$ specify the subspace in the state space that induces most change, while the patterns $\mathbf{m}^{(r)}$ specify the directions of the change applied. The connectivity matrix $J$ is defined as

$$J_{ij} = \frac{1}{N} \sum_{r=1}^{R} m_i^{(r)} n_j^{(r)}$$

and the dynamics of the $i$th neuron is given by

$$\tau \frac{dx_i}{dt} = -x_i + \sum_{j=1}^{N} J_{ij} \phi(x_j) + I_i u(t), \quad i = 1, ..., N.$$

From this follows that the activation vector $\mathbf{x} = \{x_i\}_{i=1,...,N}$ can be expressed as a linear combination of the column connectivity patterns $\mathbf{m}$ (recurrent subspace)

and the input patterns $I$ (input subspace) which span the space the dynamics live in. This is described as

$$\mathbf{x}(t) = \sum_{r=1}^{R} \kappa_r(t)\, \mathbf{m}^{(r)} + u(t)\, \mathbf{I},$$

where $\Phi$ is the hyperbolic tangent, $I$ is an input weights vector and $u$ is an input signal. The activity along the recurrent subspace can be described as a set of R internal ( (time dependent) collective variables $\kappa_r$.

### 1.2 Minimal ranks required to solve DM tasks

Debreuil et al. [2] have found that the minimal rank required for solving the *perceptual DM* task is one, whereas the *parametric PM* task requires at least a network of rank two. Apparently, there is a correlation between the number of stimuli presented and the number of ranks required. Since $\mathbf{m}$ denotes the principal direction of activity in state space (space spanned by the neurons), $\mathbf{m}$ points in the direction of the largest variance. Hence, solving the *perceptual DM* task requires only one principal component $\mathbf{m^{(1)}}$ because there is only one stimulus presented, so one eigenvector is sufficient to explain the variability that comes with the stimulus. Analogously, with two different presented stimuli in the *parametric PM* task we need an additional eigenvector to explain the additional variability coming with the second stimulus. Thus, a rank two network is needed which comes with a second principal component $\mathbf{m^{(2)}}$.

## 2 Perceptual decision making

In order to solve this task, "a unit rank network was trained to integrate a fluctuating scalar input and report whether its average was positive or negative" [2]. To this end, we created a fluctuating input given by:

$$u(t) = \begin{cases} \overline{u} + \xi(t) & , 5 \leq t \leq 45 \\ \xi(t) & , otherwise \end{cases}$$

where $\overline{u}$ is uniformly drawn from $\pm\{1, 2, 4, 8, 16\}$. The target $y$ is defined as the sign of $\overline{u}$.

### 2.1 Training the network

We trained a recurrent neural network with 128 hidden units on an input tensor with 32 trials on a single batch until it converged below approximately 5e-2, where each trial represents a stimulus duration of 75 time steps. To this end, we minimized the mean squared error using stochastic descent at each of the last 15 time steps.

In Figure 1, we plotted the loss function over the number of epochs on the left, the dynamics of a single hidden neuron of a single trial in the middle, and the output dynamics of a single trial on the right.

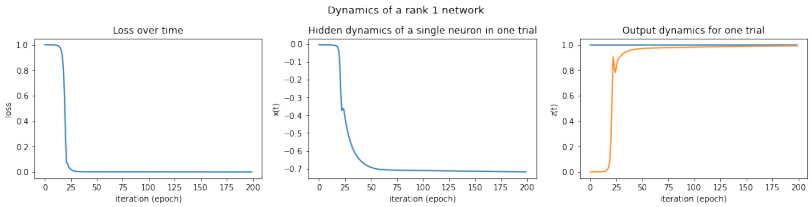



Figure 1: Training the low-rank RNN.

## 2.2 Resampling of the trained network

Within this framework, each neuron is characterized by its set of *loadings* corresponding to its input $\mathbf{I}$, readout $\mathbf{w}$ and connectivity patterns $\mathbf{m}$ and $\mathbf{n}$. This means that each neuron corresponds to a single point in loading space spanned by the corresponding patterns. Debreuil et al. [2] considered the situation in which $P$ populations of neurons correspond to $P$ Gaussian clusters in the loading space.

Two selected two-dimensional projections of the connectivity patterns in the loading space suggest a single Gaussian cluster fully characterized by the corresponding correlation matrix (Figure 2 and 3). In fact, their work revealed that a single, global Gaussian population was actually sufficient to implement the perceptual decision making task. From this correlation matrix (covariance matrix without variances) they extracted new networks that performed as good as the original trained network. This works because multivariate Gaussian mixture models (GMMs) are used which can approximate arbitrary joint distributions of loadings.

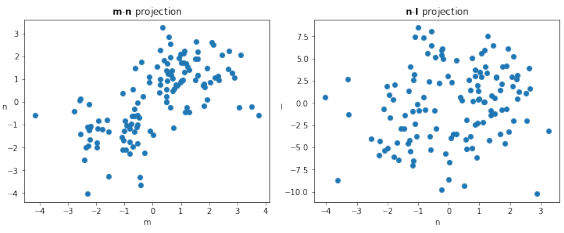



Figure 2: Selected 2D projections of the loading space.

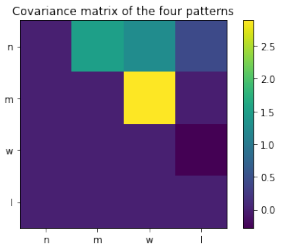



Figure 3: Covariances of the pattern loadings.

After fitting a 4D Gaussian distribution to the connectivity vectors, and then

training a network with resampled connectivity we found out that their performance was the same.

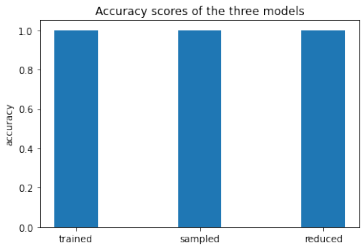



Figure 4: Accuracy scores of the trained, resampled and reduced model.

This suggests that fitting a single Gaussian cluster was actually sufficient to solve this task as we expected from the theoretical analysis.

## 2.3 Dimensionality reduction

A general approach to dimensionality reduction would be to apply PCA by computing the covariance matrix of observations of the set $\{\mathbf{x}_t \mid t \in [0, T]\}$ and extract the largest principal components and the corresponding vectors which would then span a new subspace with lower dimensionality. However, given the explanation in 1.1, we already know that the dynamics live in the space spanned by $\mathbf{I}$ and $\mathbf{m}$. We then projected the time-varying dynamics $x(t)$ onto the $\mathbf{I}$-$\mathbf{m}$ plane (Figure 5). Since the activity of the neurons in a unit rank network is given by a linear combination of $\mathbf{m}^{(1)}$ and $\mathbf{I}$ with the internal collective variables $\kappa$ and the input $\mathbf{u}(t)$, we concluded that these trajectories correspond to $\kappa$ and $\mathbf{u}(t)$ respectively.

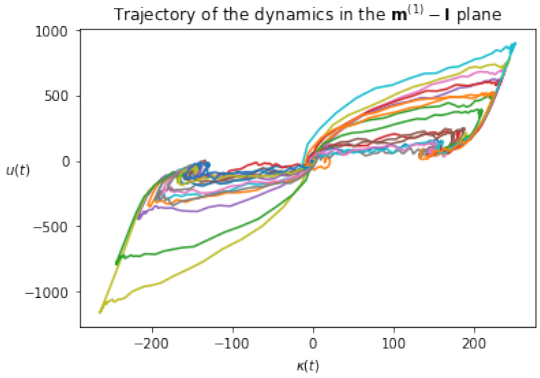



Figure 5: Dimensionality reduction.

Finally, we distilled the trained network into an equivalent one-dimensional dynamical system of the form

$$\frac{d\kappa}{dt} = -\kappa(t) + \tilde{\sigma}_{mn}\kappa(t) + \tilde{\sigma}_{nI}\nu(t),$$

and compared its performance with the previously trained networks. Apparently, this reduced network also performed with the same accuracy as the other networks (see Figure 4). By looking at the dynamics of the reduced model explicitely, the equivalent circuit indeed seems to perform the task in the sense that it is succeful in reporting whether the fluctuating input $\nu(t)$ was positive or negative. This can be well observed by looking at the third plot showing the values of the readouts that are expressed in terms of the internal and external collective variables $\nu(t)$ and $\kappa(t)$.

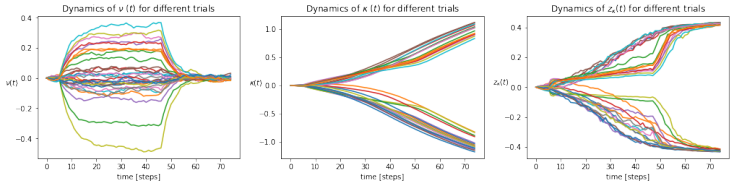



Figure 6: Collective variables

# 3 Parametric working memory

On the search of working memory mechanisms, the comparison between two scalar values is one of the most basic possible. The parametric working memory task (PWMT) was studied in monkeys. A vibration with different frequencies was applied to the fingertips sequentially. In the prefrontal cortex neuronal "discharge rates varied, during the delay period between two stimuli, as a monotonic function of the base stimulus frequency"[4]. For our analysis, we trained a rank two network on inputs consisting of two stimuli with the output being the normalized difference between these stimuli.

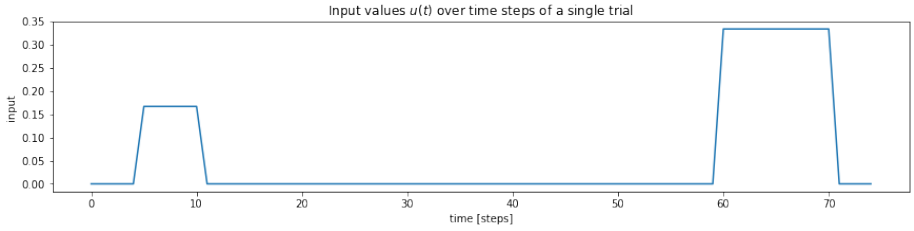



Figure 7: Input to the parametric working memory task.

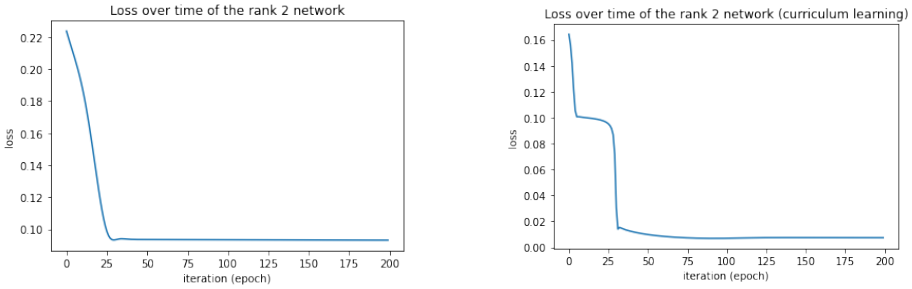



Figure 8: Convergence of the RNN with and without curriculum learning

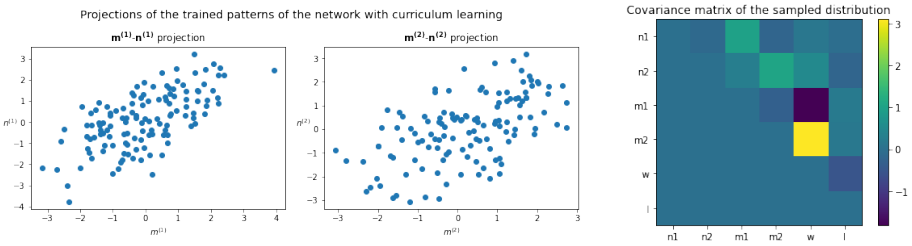



Figure 9: On the right side of the figure the covariance matrix with curriculum learning shows positive covariance of input selection patterns **n** with it's paired **m** principal activity patterns. Input weights are again not much correlated with the other parameters, whereas readout weights have salient covariances with vectors **m**.

## 3.1 Simulation of rank two networks

We trained the network until the loss converged to approximately 5e-3 using both the traditional learning, but also the curriculum learning approach by gradually increasing the delay period between the two stimuli, starting from 25 time steps. Apparently, curriculum learning gave us better results in terms of convergence (see Figure 8). We then visualized the connectivity patterns (see Figure 9) for the network trained using curriculum learning, fit a 6D Gaussian cluster and trained a new network with resampled connectivity which actually performed as good as the original trained network.

## 3.2 Dimensionality reduction

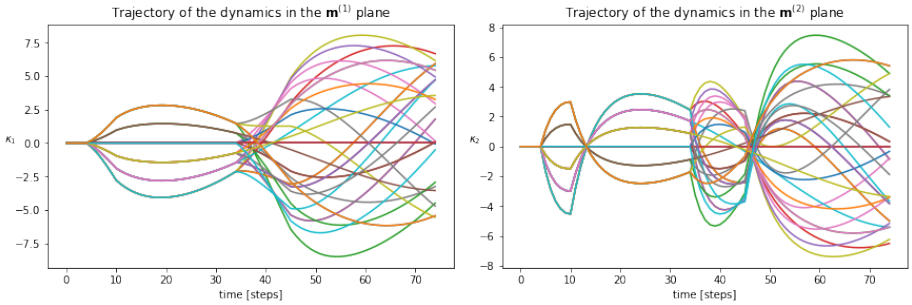
Given appropriate assumptions, the trained network can now be described by a two-dimensional equivalent circuits with dynamics given by [1]

$$\frac{d\kappa_1}{dt} = -\kappa_1(t) + \tilde{\sigma}_{m_1 n_1}\kappa_1(t) + \tilde{\sigma}_{n_1 I_1}\nu(t)$$

$$\frac{d\kappa_2}{dt} = -\kappa_2(t) + \tilde{\sigma}_{m_2 n_2}\kappa_2(t) + \tilde{\sigma}_{n_2 I_2}\nu(t)$$

In Figure 10 the trajectory of the dynamics are shown after projecting $x(t)$ onto the two principal directions $\mathbf{m^{(1)}}$ and $\mathbf{m^{(2)}}$. In Figure 11 the dynamics of the final equivalent circuit are shown. We can well observe the curriculum learning approach by looking at the dynamics of $\nu$ showing the increasing time delay.

a)




b)

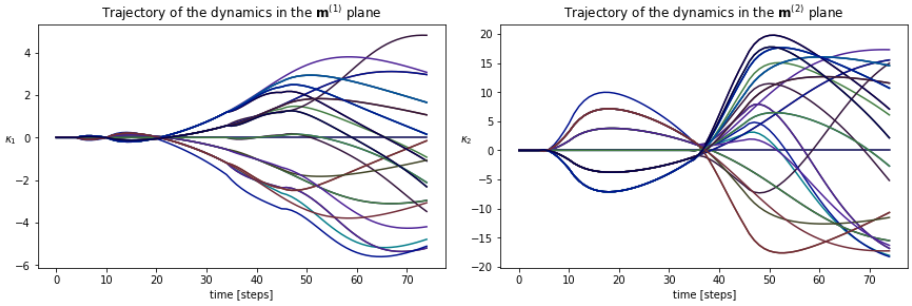



Figure 10: In the figures presented, most trajectories are non-zero from the onset of the first stimulus, and spread and follow one of the six paths. At the 35th time step of the second stimulus input they take separate paths (combinations of the first and second stimulus strength). a) Although in the dynamics projected on the second vector $\mathbf{m}^{(2)}$ orbits cross zero twice, the plots are not enough to see big topological difference between projections on $\mathbf{m}^{(1)}$ and $\mathbf{m}^{(2)}$. In the right figure the sharp absolute decrease of values around 10th time step marks stop of the input signal, wheras at the left figure the values persist until exposure to second stimulus (during curriculum learning it was at 35th time step). Again, the right figure present non smooth change during the second stimulus onset. It suggests, that the first connectivity vector plays a role in storing the information, as the second might work on comparing. b) The temporal dynamics of the reduced model looks smooth everywhere. The second pair of connectivity patterns (lower right) have low values untill the onset of second stimuli. The dynamics $C^{\infty}$, because of definition.

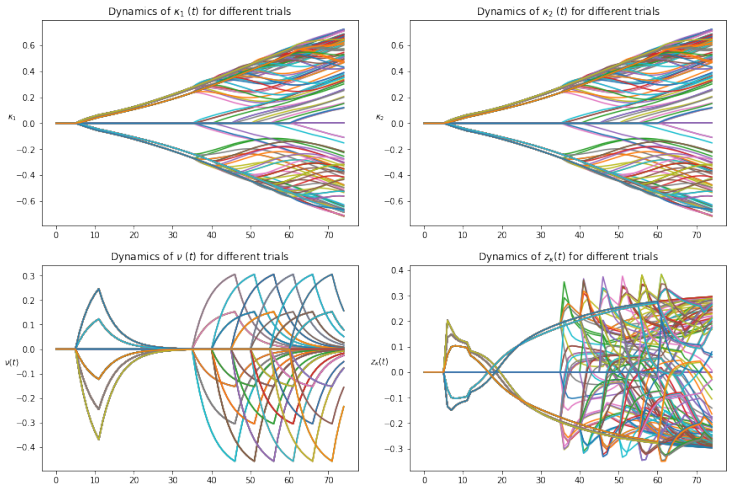



Figure 11: Dynamics of the reduced equivalent circuit.

# 4 Future work and limitations

Connecting few RNNs can be treated as a bigger RNN with subpopulations. It is easy with an assumption of the common update frequency (which is sufficient in investigating rate coded neural data) and without delay. Can the framework be expanded to systems with a delay between RNNs and different update frequency?

Another topic to investigate is the evaluation of computational complexity of this approach for advanced behavior, different rank connectivity and number of subpopulations and sizes of the network.

Analytical study of curriculum learning has confirmed, that " (...) while curriculum has no effect for standard training losses, connecting successive learning phases with a Gaussian prior (or equivalently, L2 penalty) between weights can result in a large improvement in test performance."[5]. In this project curriculum helped with interpretability of the covariance matrix (figure 9). Distinguishing class of tasks for which curriculum learning is suitable and could be automated, together with defining features of "interpretability" is an open subject. A possible approach is reduction of symmetries in a dynamical system [7]. That rises a theoretical question of its relation to the dimensionality reduction using Gaussian nature of connectivity.

The brain computer interface applications use a subset of neurons for readout. How to estimate a minimal number of non-zero values in the readout vector $\mathbf{w}$ for this framework?

# References


[1] The BCCN project description *Decision making in low-rank recurrent neural networks*

[2] Dubreuil et al., 2020

[3] Barak, Current Opinion in Neurobiology Vol. 46, 1-6 (2017), *Recurrent neural networks as versatile tools ofneuroscience research*

[4] Romo, R., Brody, C., Hernández, A. et al., Nature 399, 470–473 (1999), *Neuronal correlates of parametric working memory in the prefrontal cortex*

[5] Luca Saglietti, Stefano Sarao Mannelli, Andrew Saxe, *An Analytical Theory of Curriculum Learning in Teacher-Student Networks*

[6] Francesca Mastrogiuseppe, Srdjan Ostojic, *Linking connectivity, dynamics and computations in low-rank recurrent neural networks*

[7] Evelyne Hubert, George Labahn. *Scaling Invariants and Symmetry Reduction of Dynamical Systems*, Foundations of Computational Mathematics, Springer Verlag, 2013, 13 (4), pp.479-516.

[8] The code of the simulations here presented is available at https://github.com/Hiyeri/Low-rank-RNN