Forward Models in the Cerebellum using Reservoirs and Perturbation Learning

Katharina Schmid (katharina.schmid@s2017.tu-chemnitz.de)

Julien Vitay (julien.vitay@informatik.tu-chemnitz.de)

Fred H. Hamker (fred.hamker@informatik.tu-chemnitz.de)
Chemnitz University of Technology - Artificial Intelligence Lab, Straße der Nationen 62
Chemnitz 09107, Germany

Abstract

The cerebellum is thought to be able to learn forward models, which allow to predict the sensory consequences of planned movements and adapt behavior accordingly. Although classically considered as a feedforward structure learning in a supervised manner, recent proposals highlighted the importance of the internal recurrent connectivity of the cerebellum to produce rich dynamics, as well as the importance of reinforcement-like mechanisms for its plasticity. Based on these models, we propose a neuro-computational model of the cerebellum using an inhibitory reservoir architecture and biologically plausible learning mechanisms based on perturbation learning. The model is trained to predict the position of a simple robotic arm after ballistic movements. Understanding how the cerebellum is able to learn forward models might allow elucidating the biological basis of model-based reinforcement learning.

Keywords: Computational Neuroscience; Reservoir computing; Cerebellum; Forward models; Perturbation learning.

Introduction

Recent models have described the recurrent connectivity of the cerebellum in terms of dynamical reservoirs (Yamazaki & Tanaka, 2007; Rössert, Dean, & Porrill, 2015). Granule cells and Golgi cells indeed form a recurrent inhibitory network, actively storing information about the history of recent cortical inputs. This information is encoded in the changing activity of Granule cells and forwarded via the parallel fibres to the Purkinje cells and ultimately to output neurons in the deep cerebellar nuclei. As in reservoir computing (Jaeger, 2001; Maass, Natschläger, & Markram, 2002), the connectivity of the reservoir is left unchanged and learning takes place by adapting the connections between the reservoir and the read-out neurons. This is in agreement with classical theories of cerebellar learning, which suggest that synaptic plasticity between parallel fibres and Purkinje cells is the main mechanism of motor learning in the cerebellum (e.g. Albus, 1971; Ito, 2000).

The classical Marr-Albus-Ito hypothesis is based on supervised learning and long-term depression at parallel fibre-Purkinje cell synapses (Albus, 1971; Ito, 2000). More recently, Bouvier et al. (2018) proposed an alternative framework of cerebellar learning which borrows from reinforcement learning and relies on both long-term depression and potentiation at the synapses between parallel fibres and Purkinje cells. In this framework, climbing fibres, which reach Purkinje cells from the inferior olive, do not just provide teaching signals but are also the source of random perturbations. These perturbations are used to vary the cerebellar output and explore different responses by trial and error. A binary teaching signal indicates whether the perturbations improved or worsened the cerebellar response, adapting the synapses accordingly.

We propose a neuro-computational model combining the reservoir model of cerebellar interneurons of Rössert et al. (2015) with the perturbation learning rule for Purkinje cells proposed by Bouvier et al. (2018), in order to learn a forward model of an effector. We focus here on a simple 2-joint planar arm to demonstrate the capacity of the network to learn using biologically plausible learning rules.

Methods

The task of the forward model is to predict the next position of an end effector $(x_{\text{next}}, y_{\text{next}})$ given its previous position $(x_{\text{prev}}, y_{\text{prev}})$ and a change in joint angles $\Delta\theta_0$, $\Delta\theta_1$ representing the motor command. The positions $(x_{\text{prev}}, y_{\text{prev}})$ are obtained from the joint angles (θ_0, θ_1) (initially taken randomly in $[0, \pi]$) with a simple mechanical model of the arm (both segments have a length of 0.5). The motor commands $\Delta\theta_0$ and $\Delta\theta_1$ are sampled from $\mathcal{U}(-\frac{\pi}{18}, \frac{\pi}{18})$ to obtain $(x_{\text{next}}, y_{\text{next}})$, producing large enough movements to make the prediction problem non-linear.

Network Architecture

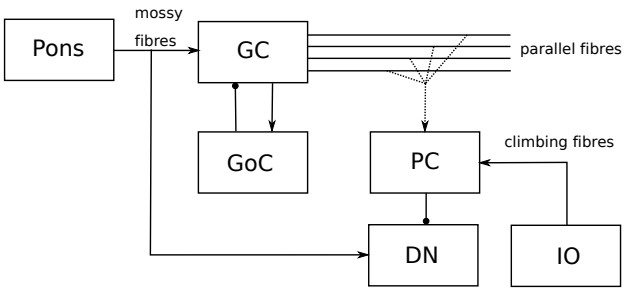

Figure 1: Network architecture: GC - granule cells, GoC - Golgi cells, PC - Purkinje cells, DN - dentate nucleus, IO - inferior olive. Edges ending with an arrow head indicate excitatory connections, edges ending in a dot indicate inhibitory connections. Dashed lines are plastic.

The architecture of the model is depicted in Figure 1. The

two population model by Rössert et al. (2015) serves as the reservoir, consisting of $N_z = 1000$ granule cells (GC) and $N_g = 100$ Golgi cells (GoC). The convergence of granule to Golgi cells is limited to $c = 10$ and the variability of excitatory weights is increased to $v_u = 4$ to improve performance (please refer to (Rössert et al., 2015) for the meaning of these parameters). The baseline of mossy-fibre activity I_{0i} is chosen randomly from $\mathcal{N}(1.2, 0.1)$. The mean of excitatory weights is set to $u = 50$ and the mean of inhibitory weights is set to $w = 0.05/\tau_w$ to keep the network dynamics close to the edge of chaos. Contrary to Rössert et al. (2015), who trained their model on a single input x, our task uses four different input signals $(x_{\text{prev}}, y_{\text{prev}}, \Delta\theta_0, \Delta\theta_1)$. Each of the granule and Golgi cells is randomly assigned a single input. Figure 2 shows the response of 20 granule cells to a random input signal presented for 20 ms. All other reservoir parameters are identical to Rössert et al. (2015).

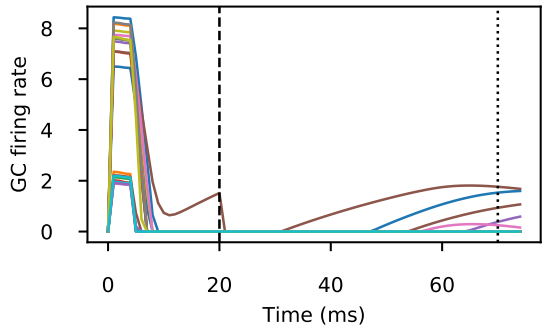

Figure 2: Activity of 20 randomly chosen granule cells for a random input sample. The dashed and dotted lines represent the end of the input presentation and the beginning of the response period.

The reservoir is read out by additional feedforward layers consisting of $N_p = 20$ Purkinje cells (PC), $N_d = 2$ projection neurons in the dentate nucleus (DN) and $N_i = 2$ neurons in the inferior olive (IO). The firing rate $p_i(t)$ of a PC is defined by:

$$p_i(t) = \left[\sum_j^{N_z} D_{ij} m_{ij} \sum_{s=1}^{t} \exp\left(-\frac{t-s}{\tau_m}\right) z_j(s-1) + \sum_j^{N_i} H_{ij} w_{IO} \sum_{s=1}^{t} \exp\left(-\frac{t-s}{\tau_{\text{IO}}}\right) l_j(s-1) \right]^+ \quad (1)$$

D and H are binary matrices describing the connectivity between the granule cells of rate $z_j(t)$, respectively IO cells of rate $l_j(t)$, and the Purkinje cells. The PC's firing rate itself is not dynamic, but each synapse integrates exponentially its inputs, leading to a similar effect. The connectivity is chosen so that 10 Purkinje cells project to the same DN neuron and receive input from the same IO neuron. The plastic weights connecting GCs and PCs are initialized with $m = [\mathcal{N}(0.1, 0.05)]^+$. The climbing fibre weights w_{IO} are all set to 1.0. τ_m and τ_{IO} are the time constants of the different synapses and are both set to 1 ms. l_j refers to the firing rate of the j-th inferior olive neuron and represents the perturbations due to climbing fibre input.

How Purkinje cells modulate cerebellar output is defined by the following equation for the firing rate $n_i(t)$ of the N_d DN neurons:

$$n_i(t) = \left[I_i(t) - \sum_j^{N_p} G_{ij} w_{PC} \sum_{s=1}^{t} \exp\left(-\frac{t-s}{\tau_{PC}}\right) p_j(s-1) \right]^+ \quad (2)$$

Like GC and GoC, DN neurons are randomly connected to a single mossy fibre input $I_i(t)$, with the push-pull mechanism described in (Rössert et al., 2015). G is a binary matrix and represents the connectivity between PC and DN neurons, as described in the previous paragraph. The corresponding connection weights w_{PC} are fixed and set to 0.1. The time constant for synaptic integration is $\tau_{PC} = 1$ ms. As the firing rate is restricted to be positive, x_{next} and y_{next} are translated and scaled to fit in the range $[\frac{I_{0i}}{4}, \frac{3I_{0i}}{4}]$. The complete neural network was implemented using ANNarchy (Vitay, Dinkelbach, & Hamker, 2015), a neural simulator for distributed rate-coded and spiking networks, with a step size of 1 ms. The code is available at `github.com/kimschmi/CerebellumForwardModel`.

Perturbation-based Learning Rule

As in Bouvier et al. (2018), each synapse between the i-th GC and the j-th PC maintains an eligibility trace $e_{ij}(t)$ according to:

$$e_{ij}(t) = e_{ij}(t-1) + \left(\sum_{s=1}^{t} \exp\left(-\frac{t-s}{\tau_m}\right) z_i(s-1) \right) \times \left(\sum_{s=1}^{t} \exp\left(-\frac{t-s}{\tau_{IO}}\right) l_k(s-1) \right) \quad (3)$$

The eligibility trace integrates the product of the presynaptic activity and the postsynaptic perturbation over time. As a consequence, only those synapses whose presynaptic neurons were active when postsynaptic perturbations occurred will be updated. In the current model, the eligibility trace does not decay over time and relies on the precise timing of perturbations. It is hypothesized that perturbations only occur during the response period, when the network's response to the input is computed. An alternative would be to consider a decaying eligibility trace, which would however limit the response period to a short time before the error feedback.

As the information about the network error becomes available, the weight updates are computed by:

$$\Delta m_{ij} = -\eta\, c\, e_{ij} \quad (4)$$

$\eta = 2e{-}4$ is the learning rate, e_{ij} the eligibility trace and c signals whether the error for the current input-output pair improved or got worse relative to previous trials. This binary

signal is given by:

$$c = \text{sign}(\varepsilon^\mu - I^\mu) \tag{5}$$

The current error ε^μ for an input μ is the Euclidean distance between the network output and the target value. The average of recent errors for a specific input pattern I^μ is estimated by a linear combination of the GC outputs $I^\mu = \sum_i v_i z_i(t)$, where $z_i(t)$ is the firing rate of the i-th GC and the coefficient v_i is updated at the end of each trial according to:

$$\Delta v_i = \eta \; c \sum_{t=70}^{75} z_i(t) \tag{6}$$

The firing rate of the granule cells is averaged over the response period $[70, 75]$ (see next section) and $\eta = 2e{-}4$ is the learning rate. The absolute value of both weight updates Δm and Δv is not allowed to exceed 0.5 to ensure stability.

In the original version by Bouvier et al. (2018), the synaptic weights between parallel fibres and Purkinje cells are not restricted to be positive, which they explain by the fact that the GC firing rate may be relayed to Purkinje cells via inhibitory interneurons. During learning, synaptic weights can therefore change their sign, which could correspond to the creation of new inhibitory synapses and the pruning of existing excitatory synapses or vice versa. This form of structural plasticity cannot be explained by current observations of long-term depression and long-term potentiation at these synapses. We restrict here the weights to positive values only.

Training Procedure

The network is presented with the four input signals x_{prev}, y_{prev}, $\Delta\theta_0$ and $\Delta\theta_1$ for 20 ms. After a delay of 50 ms, the network response is read out. During this response period of 5 ms, the Purkinje cells receive random perturbations that vary the cerebellar output. Perturbations are generated randomly and independently by each IO neuron with a mean rate of 50 Hz (the probability that a DN neuron receives a perturbation during the response period is 0.25) and an amplitude of 0.1. Finally, the network response is evaluated, compared with the desired positions, and the parallel fibre-Purkinje cell weights are updated. Between the presentation of different input samples, the firing rate of granule cells and Golgi cells and the results of synaptic integration are reset to values chosen randomly from $\mathcal{U}(0.0, 0.1)$. The network is trained on a set of 5,000 random samples for 2,000 epochs and its performance is evaluated on a test set of another 5,000 random samples.

Results

Figure 3 shows that the mean remaining error between the prediction and the arm's target position decreases rapidly during learning. Figure 4 shows the time course of the activity of the two projection neurons when the network is presented with a random test sample after learning.

Figure 5 illustrates how the corresponding predicted arm position evolves during learning for that particular movement.

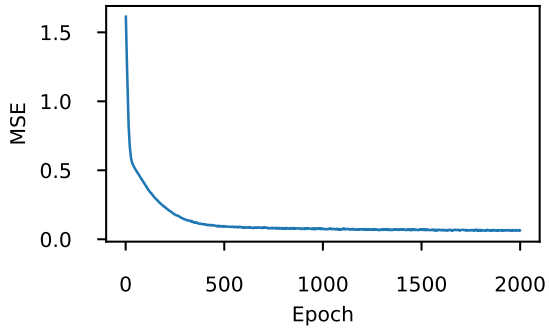

Figure 3: Development of the prediction error during perturbation learning with positivity constraint.

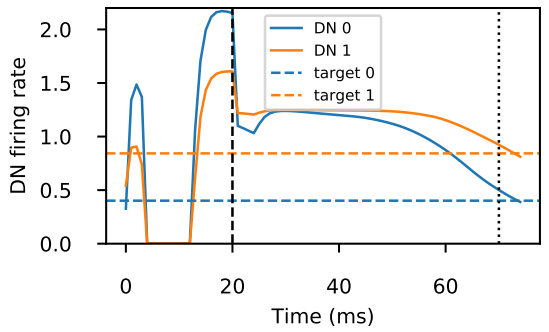

Figure 4: Activity of DN projection neurons for a random input sample. Their target value is represented by horizontal lines.

The initial prediction is completely wrong, as the weights are initialized randomly, reaches quite quickly a mean target position around $(0, 0.5)$ and finally converges quite slowly to the target value (note the remaining error). The same pattern can be observed for most learned movements, as can be seen in Figure 6, where the variance of the distance between this mean position and the predicted positions first decreases sharply, before increasing again. Eventually, learning converges to a mean Euclidean distance of 0.079 over both the training and test sets. Removing the positivity constraint on the weights between the parallel fibres and the PC slightly reduces the mean error to 0.069 on both sets.

Figure 7 visualizes the distribution of the network prediction errors in the arm's workspace (test set). The accuracy of the predictions is generally better around the mean position $(0.0, 0.5)$ than in extreme arm positions, indicating the non-linearity of the learning problem.

Discussion

The proposed model combines the recurrent dynamics of the GC-GoC excitatory-inhibitory network proposed by Rössert et al. (2015) with the perturbation-based learning rule for parallel fibres-PC synapses proposed by Bouvier et al. (2018). The model is able to learn a simple non-linear prediction task on a

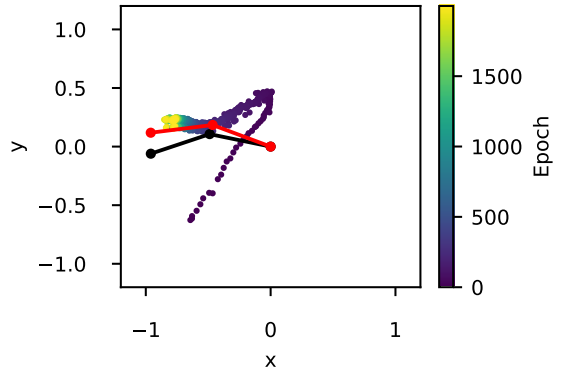

Figure 5: Development of the predicted arm position for a particular movement during training. The black arm indicates the initial position, the red arm indicates the arm position after the movement. The colored dots show the predicted positions during learning.

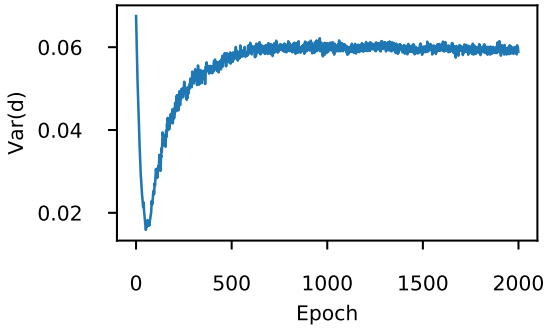

Figure 6: Evolution of the variance of the distance d between the mean position of the arm and the predictions during training, computed over 500 random training samples. This indicates that the network first tries to make all predictions close to the mean position and only later learns individual predictions.

2D simulated arm, although still imprecisely. Contrary to the classical supervised approach requiring complete error signals, the model learns from a binary teaching signal indicating whether the prediction error has improved compared to baseline performance. This allows to learn forward models with a cerebellar model: the IO mainly receives low-level motor and proprioceptive information, so it can only drive supervised learning of inverse models (motor adaptation). For supervised forward models, the IO would need to compare cortical sensory representations in order to compute the teaching signal, what seems to be a very challenging task for such a small nucleus. By relying on a much simpler reinforcement-like teaching signal, the proposed model could learn forward models even if the predicted sensory space is high dimensional.

Many aspects of the proposed model need to be further studied, such as its precision, its use for high dimensional sensory spaces or longer prediction delays. An efficient and biologically realistic model of the cerebellum for the acquisition of forward sensory models would be an important step for the study of many cognitive functions, including model-based reinforcement learning, planning, body awareness or the sense of agency. The integration of this model into broader neural architectures involving the cerebral cortex and the basal ganglia might allow to both understand better the brain at the systems-level and create adaptive cognitive agents.

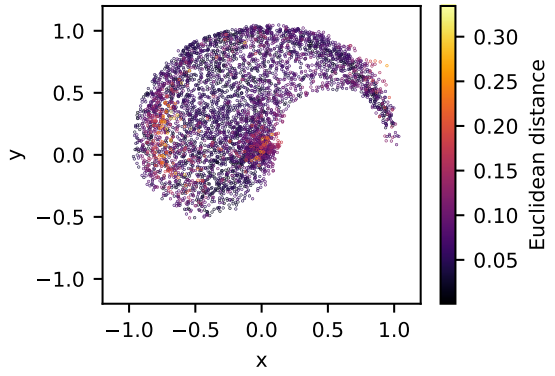

Figure 7: Distribution of the prediction error in the arm's workspace. The color indicates the Euclidean distance between the target position and the network's response after training.

Acknowledgments

This work was partially supported by the DFG priority program "The Active Self" HA2630/12-1.

References

Albus, J. S. (1971). A theory of cerebellar function. *Mathematical Biosciences*, *10*(1-2), 25–61.

Bouvier, G., Aljadeff, J., Clopath, C., Bimbard, C., Ranft, J., Blot, A., ... Barbour, B. (2018). Cerebellar learning using perturbations. *eLife*, *7*, e31599.

Ito, M. (2000). Mechanisms of motor learning in the cerebellum. *Brain research*, *886*(1-2), 237–245.

Jaeger, H. (2001). *The "echo state" approach to analysing and training recurrent neural networks* (Tech. Rep.).

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, *14*(11), 2531–2560.

Rössert, C., Dean, P., & Porrill, J. (2015). At the edge of chaos: how cerebellar granular layer network dynamics can provide the basis for temporal filters. *PLoS computational biology*, *11*(10), e1004515.

Vitay, J., Dinkelbach, H., & Hamker, F. (2015). ANNarchy: a code generation approach to neural simulations on parallel hardware. *Frontiers in Neuroinformatics*, *9*, 19.

Yamazaki, T., & Tanaka, S. (2007). The cerebellum as a liquid state machine. *Neural Networks*, *20*(3), 290 - 297.