MCC Modular Cerebellar Circuit: User Manual [*]

Marie Claire Capolei, Silvia Tolu, Henrik Hautop Lund

{macca,stolu,hhl}@elektro.dtu.dk

Automation and Control Group, Department of Electrical Engineering,

Technical University of Denmark, Denmark

Contents

[*]This work has received funding from the EU-H2020 Framework Program for Research and Innovation under the specific grant agreement No. 785907 (Human Brain Project SGA2)

1 Introduction

The Modular Cerebellar Circuit (***MCC***) is an algorithm that learns online a pattern trough incremental learning, and elaborates an adaptive correction for the control input in order to reduce the error signal. The **MCC** design replicates the functionality, learning, modularity, and synaptic organization of the biological cerebellar-circuit, a neural structure located at the back of the brain, through the combination of machine learning, artificial neural networks, and computational neuroscience techniques. The network weights are defined by non-linear and multidimensional learning functions that mimic the cerebellar synaptic plasticities, as proposed by [1], [2]. Moreover, the biologically plausible weighting kernel together with the layered structure of the cerebellar network result beneficial to constrains the effects of perturbations and nonlinearities.

The **MCC** has been exploited to control a robotic plant, in order to test specific hypothesis regarding the cerebellum, and to improve the performance of the complex robotic system while it is interacting with unknown and dynamic environment [3], [4].

The structure of the user manual is as follows: in section 1.1 we shortly describe the canonical cerebellar micro-circuit; in section 1.2, the mathematical models employed in the network are presented; in section 2 the elements of the python class are listed and explained. The manual concludes with an example of script file.

1.1 The Cerebellar Canonical Micro-circuit

The cerebellum is constituted of several micro-zones that plausibly correspond to the minimal *ulm unit learning machine* (Fig.1.**a**) [5]. Each *ulm* presents similar internal micro-circuitry, but it differs from the others in terms of external connectivity. There are two main type of axons that connect each *ulm* to the outside: the MF *mossy fibers* (in magenta Fig.1.**a**), which project signals regarding the position, velocity and direction of the limbs movements [6]; the *climbing fibers* (in red), that project from the IO *inferior olive nucleus* the signal encoding the error [7], [8]. These axons transmits the information to two main groups of cells: the Gr *granule cells*, that in Marr's opinion encode combinations of mossy fibers inputs [9]; the PC *Purkinje cells* (in green Fig.1.**a**), that modulated by the inferior olive axon and excited by the PF *parallel fibers* (in violet) projecting from the granule cells, they influence the activity of the DCN *deep cerebellar nuclei* (in blue). The DCN is inhibited by the PC and excited by both the IO and MF, and it is responsible for the final processing of the signal that is sent outside the cerebellar circuit.

1.2 The Modular Cerebellar-like Circuit

The neural network structure is divided into separated modules (Fig. 1.**b**), or namely uml *Unit Learning Machine* [11], [12]. Assuming a robot plant composed by N controllable objects, each uml is specialized on the n-th controlled object (where $n = 1, ..., N$), or rather the DCN deep cerebellar nuclei output of the n-th uml will be the cerebellar contribution on the n-th object. The uml itself is separated into M sub-modules which represent the ccm *canonical cerebellar microcircuit*. Each ccm is specialized with respect to a specific feature

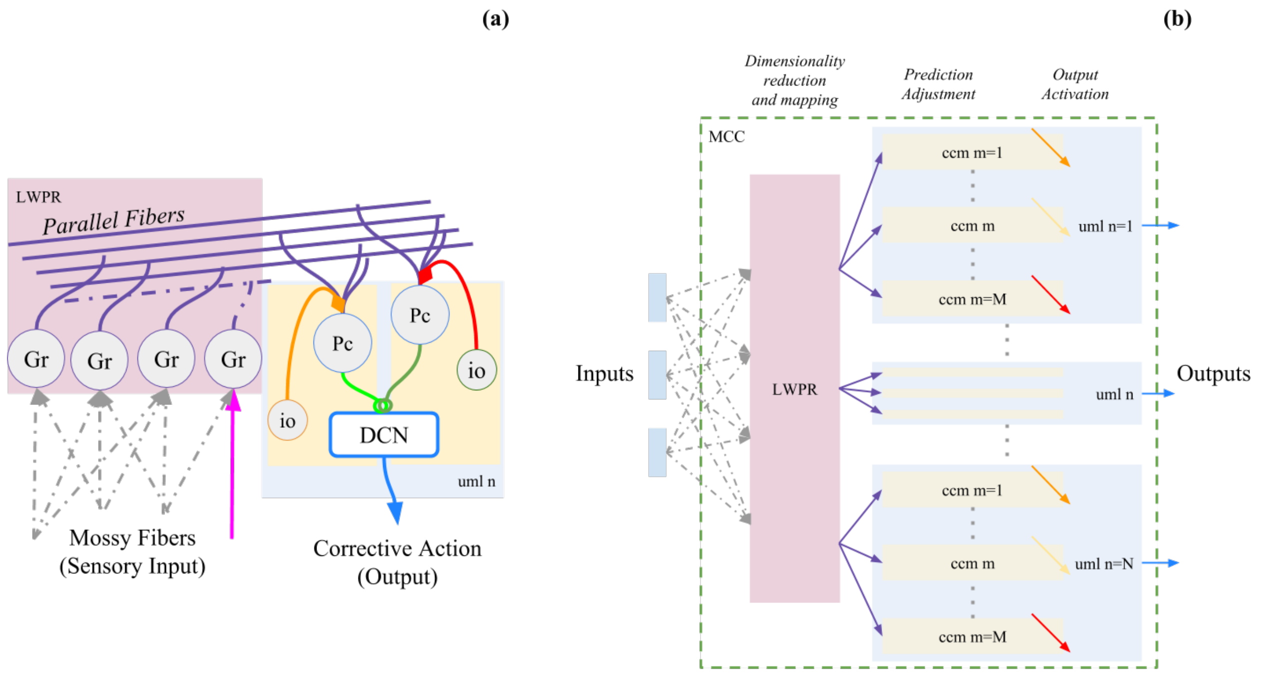

Figure 1: **a)** Proposed simplified cerebellar-like circuit in analogy with [10], **b)** neural network structural partition.

describing the behavior of the n-th controlled object, such as position and velocity. The Modular Cerebellar Circuit (MCC) is composed by the combination of all umls and the structures specialized on the dimensionality reduction, and mapping of the sensory information.

Hereafter for the sake of simplicity, the variable x generally recalls the signals propagating across the network, and w generally recalls the network weights.

In the Neural Network details (Fig. 2), the **MF** mossy fibers (in magenta) transmit the $\mathbf{x}^{mf}$ information about the current and reference states of all the controlled objects to the **GR** granular layer (in purple),

$$\mathbf{x}^{mf}_{2N\times 1}(t) = \begin{bmatrix} x^{mf}_1(t) \\ \dots \\ x^{mf}_{2N}(t) \end{bmatrix}. \tag{1}$$

The granular layer is the circuit area committed to the mapping of the mossy fibers signals, and to the prediction of the next control output given the current sensory input [9], [13]. As proposed by [11], [12], the granular layer is artificially represented by the Locally Weighted Projection Regression algorithm (LWPR) [14]. The LWPR resulted an efficient method for the fast on-line approximation of non-linear functions in high dimensional space. To replicate the efference copy theory [15], [16], the LWPR uses as teaching signal a copy of the x^{tot}_n input sent to each controlled object in order to on-line create, and train G local linear models, or namely G Gr_g granule cells (where g=1,...,G). These models are employed by the algorithm to make G $\hat{x}^{gr}_{n,g}$ local predictions of the x^{tot}_n outgoing signal. The $\hat{x}^{pf}_n$ final output of the granular layer (in violet Fig. 2), which corresponds to the signal transmitted

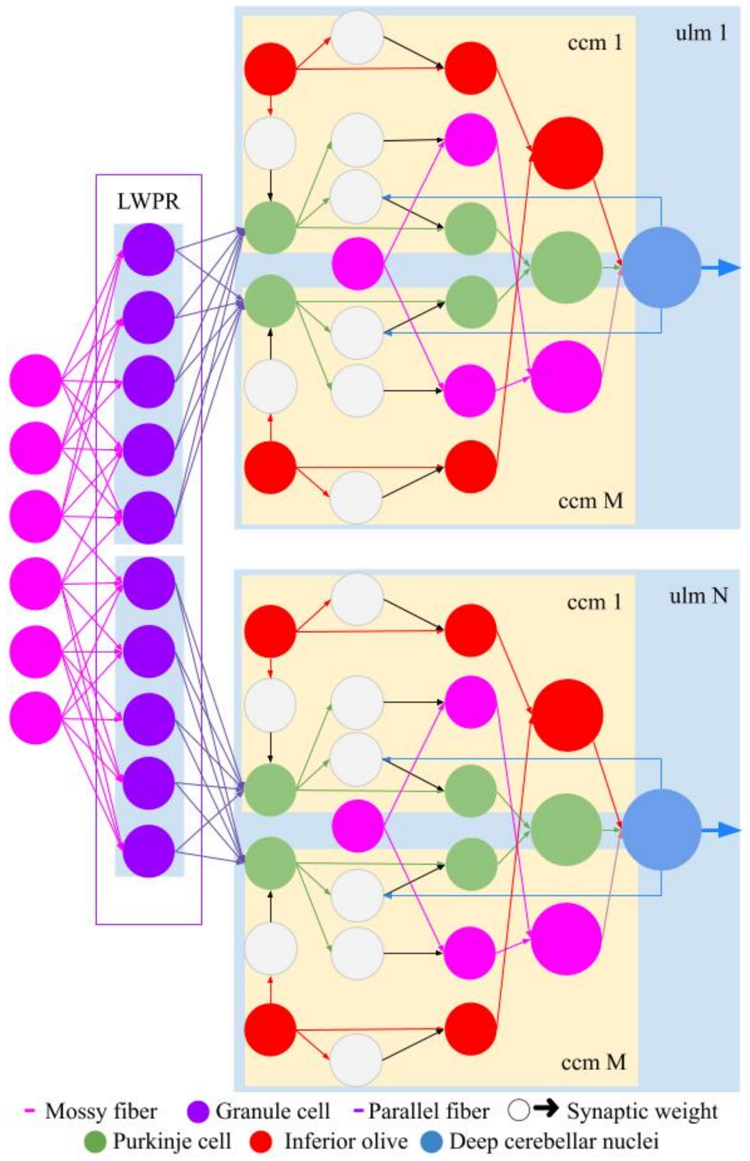

Figure 2: Details of the proposed simplified cerebellar-like neural network.

by the PF parallel fibers, is the weighted mean of all the Gr_g linear models,

$$\hat{x}_n^{pf} = \frac{\sum_{g=1}^{g=G} w_{n,g}^{gr} \cdot \hat{x}_{n,g}^{gr}}{\sum_{g=1}^{g=G} w_{n,g}^{gr}}. \tag{2}$$

where $w_{n,g}^{gr}$ and $\hat{x}_{n,g}^{gr}$ are defined in [14].

The influence of the n-th PF parallel fiber on the M PC Purkinje cells (PF-PC connection) [2], it is modulated by the $\mathbf{w}_n^{pf-pc}$ synaptic weights (in green Fig. 2),

$$\mathbf{w}_n^{pf-pc}(t) = \begin{bmatrix} w_{n,1}^{pf-pc}(t, x_{n,1}^{io}(t)) \\ \dots \\ w_{n,M}^{pf-pc}(t, x_{n,M}^{io}(t)) \end{bmatrix}, \tag{3}$$

that are function of the $\mathbf{x}_n^{io}$ inferior olive signals (in red Fig. 2),

$$\mathbf{x}_n^{io}(t) = \begin{bmatrix} x_{n,1}^{io}(t) \\ ... \\ x_{n,M}^{io}(t) \end{bmatrix} = \begin{bmatrix} \tilde{e}_{n,1}(t) \\ ... \\ \tilde{e}_{n,M}(t) \end{bmatrix}, \tag{4}$$

where $\tilde{\mathbf{e}}_n(t)$ are the normalized errors performed by each controlled object. The $\mathbf{x}_n^{pc}$ output signal of each **PC** Purkinje cell (in green Fig. 2) corresponds to the modulated $\hat{\mathbf{x}}_n^{pf}$ prediction (2),

$$\mathbf{x}_n^{pc}(t) = \begin{bmatrix} x_{n,1}^{pc}(t) \\ ... \\ x_{n,M}^{pc}(t) \end{bmatrix}, \tag{5}$$

where,

$$x_{n,m}^{pc}(t) = w_{n,m}^{pf-pc}(t, x_{n,m}^{io}(t)) \cdot \hat{x}_n^{pf}(t). \tag{6}$$

Respect to [11], [12], both the x^{io} inferior olive, the x^{pf} parallel fibers, and the x^{pc} Purkinje cells signals are reformulated:

- the inferior olive transmits the error signals instead of the control input [17];
- the x^{pf} is represented by the final LWPR prediction and not by the linear combination of the network weights;
- the x^{pc} is the result of a biologically plausible learning rule function of the error (4), instead of the direct proportion of the error signal.

The PC Purkinje cells signals transmitted to the DCN deep cerebellar nuclei (in green Fig.2),

$$\mathbf{x}_n^{pc-dcn}(t) = \begin{bmatrix} x_{n,1}^{pc-dcn}(t) \\ ... \\ x_{n,M}^{pc-dcn}(t) \end{bmatrix}, \tag{7}$$

are scaled by the $\mathbf{w}_n^{pc-dcn}$ synaptic weights [2], which are function of both the PC and DCN activities,

$$x_{n,m}^{pc-dcn} = w_{n,m}^{pc-dcn}(t, x_{n,m}^{pc}, \Delta x_n^{dcn}) \cdot x_{n,m}^{pc}. \tag{8}$$

In the proposed scheme, the MF mossy fibers (in magenta Fig.2) project the modulated copy of the $\mathbf{x}_n^{tot}(t-1)$ latest input sent to the controlled object to the DCN deep cerebellar nuclei,

$$\mathbf{x}_n^{mf-dcn}(t) = \begin{bmatrix} x_{n,1}^{mf-dcn}(t) \\ ... \\ x_{n,M}^{mf-dcn}(t) \end{bmatrix}, \tag{9}$$

these signals are highly influenced by the PC Purkinje cells activity [2],

$$x_{n,m}^{mf-dcn} = w_{n,m}^{mf-dcn}(t, x_{n,m}^{pc}) \cdot x_n^{tot}(t-1). \tag{10}$$

The IO inferior olive input to the DCN deep cerebellar nuclei,

$$\mathbf{x}_n^{io-dcn}(t) = \begin{bmatrix} x_{n,1}^{io-dcn}(t) \\ ... \\ x_{n,M}^{io-dcn}(t) \end{bmatrix}. \tag{11}$$

is determined by the IO activity itself (4) [18],

$$x_{n,m}^{io-dcn} = w_{n,m}^{io-dcn}(t, io_{n,m}) \cdot io_{n,m}. \tag{12}$$

The Δx_n^{dcn} final corrective action of each uml unit learning machine (in blue Fig.2) is function of the excitatory activities of the MF mossy fibers and the IO inferior olive, combined with the inhibitory action of the Purkinje cells,

$$\Delta x_n^{dcn} = +\, \kappa(\;\; \kappa(\sum_{m=1}^{M} x_{n,m}^{mf-dcn}) + \kappa(\sum_{m=1}^{M} x_{n,m}^{io-dcn}) - \kappa(\sum_{m=1}^{M} x_{n,m}^{pc-dcn}) \;\;), \tag{13}$$

where $\kappa(y)$ is the nonlinear activation function defined as,

$$\kappa(y) = \frac{2}{1+e^{-2y}} - 1\ . \tag{14}$$

2 Elements of the MCC Class

2.1 Necessary variables

These variables need to be initialized in the script.

n_uml (*integer*)
N number of unit learning machine (Section 1.2).

n_ccm (*integer*)
M number of local specialized units, i.e. cerebellar canonical circuit (Section 1.2).

n_input_mf (*integer*)
2N number of sensory signals input through the MF mossy fibers (Section 1.2)).

2.2 Optional Variables

Calling these variable in the script will override their default value.

name (*string*, Default = 'Cerebellum')
Name of the MCC Modular Cerebellar Circuit.

rfs_print (*integer*, Default = 1)
1 (0) to enable (disable) the visualization of the G number of local linear models created by the LWPR.

signal_normalization (*integer*, Default = 1)
1 (0) to (not) normalize the signal.

PC_on (*bool*, Default = True)
True (False) to (not) include the Purkinje cells contribution to the Deep Cerebellar nuclei.

IO_on (*bool*, Default = True)
True (False) to (not) include the inferior olive contribution to the Deep Cerebellar nuclei.

MF_on (*bool*, Default = True)
True (False) to (not) include the mossy fibers contribution to the Deep Cerebellar nuclei.

2.2.1 Debug Variables

debug_update (*integer*, Default = 0)
1 (0) to enable (disable) the print of the information about the update process.

debug_prediction (*integer*, Default = 0)
1 (0) to enable (disable) the print of the information about the prediction process.

2.3 Signals

x_MF (*float 1-dimensional array*, $1 \times n\_input\_mf$)

Mossy fibers sensory signals (Section 2.4.3 Eq.(1)),

$$x\_MF = \mathbf{x}^{mf} = \left[x_1^{mf}(t),\ \dots\ ,\ x_{n\_input\_mf}^{mf}\right].$$

x_MF_teach (*float 1-dimensional array*, $1 \times n\_uml$)

$\mathbf{x}_n^{tot}(t-1)$ signal mapped in the granular layer, e.g. signal learned by the LWPR,

$$x\_MF\_teach = \mathbf{x}^{tot} = \left[x_1^{tot}(t-1),\ \dots\ ,\ x_{n\_uml}^{tot}(t-1)\right].$$

x_MF_DCN (*float 2-dimensional array*, $n\_uml \times n\_ccm$)

$\mathbf{x}^{mf-dcn}$ mossy fibers contribution to the deep cerebellar nuclei activity (Section 2.4.3 Eq.(9)),

$$x\_MF\_DCN = \mathbf{x}^{mf-dcn} = \left[\left[x_{uml_1,ccm_1}^{mf-dcn}, \dots, x_{uml_1,ccm_M}^{mf-dcn}\right], \dots, \left[x_{uml_N,ccm_1}^{mf-dcn}, \dots, x_{uml_N,ccm_M}^{mf-dcn}\right]\right].$$

x_PF (*float 1-dimensional array*, $1 \times n\_output\_pf$)

Signal predicted by the LWPR (Section 2.4.3 Eq.(2)),

$$x\_PF = \hat{\mathbf{x}}^{pf} = \left[\hat{x}_1^{pf},\ \dots\ ,\ \hat{x}_{n\_uml}^{pf}\right].$$

x_PC (*float 2-dimensional array*, $n\_uml \times n\_ccm$)

$\mathbf{x}^{pc}$ Purkinje cells prediction modulated by the Inferior Olive (Section 2.4.3 Eq.(5)),

$$x\_PC = \mathbf{x}^{pc} = \left[\left[x_{uml_1,ccm_1}^{pc}, \dots, x_{uml_1,ccm_M}^{pc}\right], \dots, \left[x_{uml_N,ccm_1}^{pc}, \dots, x_{uml_N,ccm_M}^{pc}\right]\right].$$

x_PC_norm (*float 2-dimensional array*, $n\_uml \times n\_ccm$)

Normalized Purkinje cells contribution.

x_PC_DCN (*float 2-dimensional array*, $n\_uml \times n\_ccm$)

$\mathbf{x}^{pc-dcn}$ Purkinje cells contribution to the deep cerebellar nuclei activity (Section 2.4.3 Eq.(7)),

$$x\_PC\_DCN = \mathbf{x}^{pc-dcn} = \left[\left[x_{uml_1,ccm_1}^{pc-dcn}, \dots, x_{uml_1,ccm_M}^{pc-dcn}\right], \dots, \left[x_{uml_N,ccm_1}^{pc-dcn}, \dots, x_{uml_N,ccm_M}^{pc-dcn}\right]\right].$$

x_IO (*float 2-dimensional array*, $n\_uml \times n\_ccm$)

$\mathbf{x}^{IO}$ inferior olive signals (Section 2.4.3 Eq.(4)),

$$x\_IO = \mathbf{x}^{io} = \left[\left[x_{uml_1,ccm_0}^{io}, \dots, x_{uml_0,ccm_M}^{IO}\right], \dots, \left[x_{uml_N,ccm_0}^{IO}, \dots, x_{uml_N,ccm_M}^{IO}\right]\right].$$

x_IO_DCN (*float 2-dimensional array*, $n\_uml \times n\_ccm$)

$\mathbf{x}^{io-dcn}$ inferior olive contribution to the deep cerebellar nuclei activity(Section 2.4.3 Eq.(11)),

$$x\_IO\_DCN = \mathbf{x}^{io-dcn} = \left[\left[x_{uml_1,ccm_1}^{io-dcn}, \dots, x_{uml_1,ccm_M}^{io-dcn}\right], \dots, \left[x_{uml_N,ccm_1}^{io-dcn}, \dots, x_{uml_N,ccm_M}^{io-dcn}\right]\right].$$

x_DCN (*float 1-dimensional array*, $1 \times n\_uml$)

$\Delta\mathbf{x}^{dcn}$ deep cerebellar nuclei output signal (Section 2.4.3 Eq.(13)),

$$x\_DCN = \Delta\mathbf{x}^{dcn}\left[\Delta x_1^{dcn},\ \dots\ ,\ \Delta x_{n\_uml}^{dcn}\right].$$

2.3.1 Definition of the Signals Boundaries

range_symmetry (*2-dimensional array*, 1×2, Default = 1)
Matrix describing if the boundaries of the inferior olive signal and the deep cerebellar nuclei are 1 (0) symmetric (asymmetric),

$$range\_symmetry = \left[symmetry^{io}, symmetry^{dcn}\right].$$

range_IO (*3-dimensional array*, $n\_uml \times n\_cmm \times 2$, Default = $[0., \pi]$)
Matrix describing the boundaries of the inferior olive signal per each ccm,

$$range\_IO = [\left[\left[min_{ccm_0,uml_0}, max_{ccm_0,uml_0}\right], ..., \left[min_{ccm_M,uml_0}, max_{ccm_M,uml_0}\right]\right], ..., \left[\left[min_{ccm_0,uml_N}, max_{ccm_0,uml_N}\right], ..., \left[min_{ccm_M,uml_N}, max_{ccm_M,uml_N}\right]\right]]$$

.

range_signal (*2-dimensional array*, $n\_uml \times 2$, Default = $[0., \pi]$)
Matrix describing the boundaries of the deep cerebellar nuclei signal per each uml,

$$range\_signal = \left[\left[min_{uml_0}, max_{uml_0}\right], ..., \left[min_{uml_N}, max_{uml_N}\right]\right]$$

.

2.4 Synaptic Plasticity

2.4.1 Granular layer: the Locally Weighted Projection Regression Algorithm

Parameters described in the Locally Weighted Projection Regression (LWPR) [14] user manual [1]. Calling these variable in the script before the ***create_model()*** function will override their default value.

init_D_gr (*float*, Default = 0.5)
Initial values of distance metric.

init_alpha_gr (*float*, Default = 100)
Distance Metric initial learning rate.

w_gen_gr (*float*, Default = 0.4)
Weight activation threshold.

diag_only_gr (*bool*, Default = bool(1))
Diagonal Distance Metric.

update_D_gr (*bool*, Default = bool(0))
Distance Metric receptive fields update.

meta_gr (*bool*, Default = bool(0))
Distance Metric second order adaptation.

[1]Locally Weighted Projection Regression web-site:
https://homepages.inf.ed.ac.uk/svijayak/software/LWPR/

init_lambda_gr (*float*, Default = 0.9)
Initial forgetting factor (commented at the moment).

tau_lambda_gr (*float*, Default = 0.5)
Annealing constant for the forgetting factor (commented at the moment).

final_lambda_gr (*float*, Default = 0.995)
Final forgetting factor (commented at the moment).

w_prune_gr (*float*, Default = 0.95)
Weight prune threshold.

meta_rate_gr (*float*, Default = 0.3)
Second order learning rate (commented at the moment).

add_threshold_gr (*float*, Default = 0.95)
Regression threshold (commented at the moment).

kernel_gr (*string*, Default = 'Gaussian')
Receptive field activation function (commented at the moment).

2.4.2 Synaptic weights

w_gr (*float N-dimensional array*, *n_uml* $\times$ *G*)
w^{gr} synaptic weight of the granular cells. The dimension depends on the G number of local linear models created by the LWPR (Section 2.4.3 Eq.(2)).

w_pf_pc (*float 2-dimensional array*, *n_uml* $\times$ *n_ccm*)
w^{pf-pc} synaptic weight parallel fibers-Purkinje cells connection (Section 2.4.3 Eq.(3)),

$$w\_pf\_pc = \mathbf{w}^{pf-pc} = \left[\left[w^{pf-pc}_{uml_1,ccm_1}, ..., w^{pf-pc}_{uml_1,ccm_M}\right], ..., \left[w^{pf-pc}_{uml_N,ccm_1}, ..., w^{pf-pc}_{uml_N,ccm_M}\right]\right],$$

the weight increment is store in the variable **delta_w_pf_pc**.

w_pc_dcn (*float 2-dimensional array*, *n_uml* $\times$ *n_ccm*)
w^{pc-dcn} synaptic weights Purkinje cells-deep cerebellar nuclei connection (Section 2.4.3 Eq.(8)),

$$w\_pc\_dcn = \mathbf{w}^{pc-dcn} = \left[\left[w^{pc-dcn}_{uml_1,ccm_1}, ..., w^{pc-dcn}_{uml_1,ccm_M}\right], ..., \left[w^{pc-dcn}_{uml_N,ccm_1}, ..., w^{pc-dcn}_{uml_N,ccm_M}\right]\right],$$

the weight increment is store in the variable **delta_w_pc_dcn**.

w_io_dcn (*float 2-dimensional array*, *n_uml* $\times$ *n_ccm*)
w^{io-dcn} synaptic weight inferior olive-deep cerebellar nuclei connection (Section 2.4.3 Eq.(12)),

$$w\_io\_dcn = \mathbf{w}^{io-dcn} = \left[\left[w^{io-dcn}_{uml_1,ccm_1}, ..., w^{io-dcn}_{uml_1,ccm_M}\right], ..., \left[w^{io-dcn}_{uml_N,ccm_1}, ..., w^{io-dcn}_{uml_N,ccm_M}\right]\right],$$

the weight increment is store in the variable **delta_w_io_dcn**.

w_mf_dcn (*float 2-dimensional array, n_uml × n_ccm*)

w^{mf-dcn} synaptic weight mossy fibers-deep cerebellar nuclei connection (Section 2.4.3 Eq.(9)),

$$w\_mf\_dcn = \mathbf{w}^{mf-dcn} = \left[\left[w^{mf-dcn}_{uml_1,ccm_1}, ..., w^{mf-dcn}_{uml_1,ccm_M}\right], ..., \left[w^{mf-dcn}_{uml_N,ccm_1}, ..., w^{mf-dcn}_{uml_N,ccm_M}\right]\right],$$

the weight increment is store in the variable **delta_w_mf_dcn**.

2.4.3 Learning parameters

alpha_PF_PC (*float 2-dimensional array, n_uml × n_ccm*, Default = 1.)

Decaying factor of the parallel fibers-Purkinje cells connection [2],

$$alpha\_PF\_PC = \left[\left[\alpha^{pf-pc}_{uml_1,ccm_1}, ..., \alpha^{pf-pc}_{uml_1,ccm_M}\right], ..., \left[\alpha^{pf-pc}_{uml_N,ccm_1}, ..., \alpha^{pf-pc}_{uml_N,ccm_M}\right]\right].$$

ltp_PF_PC_max (*float 2-dimensional array, n_uml × n_ccm*, Default = $10.^{-3}$)

Maximum long term potentiation factor of the parallel fibers-Purkinje cells connection [2],

$$ltp\_PF\_PC\_max = \left[\left[ltp^{pf-pc}_{uml_1,ccm_1}, ..., ltp^{pf-pc}_{uml_1,ccm_M}\right], ..., \left[ltp^{pf-pc}_{uml_N,ccm_1}, ..., ltp^{pf-pc}_{uml_N,ccm_M}\right]\right].$$

ltd_PF_PC_max (*float 2-dimensional array, n_uml × n_ccm*, Default = $10.^{-3}$)

Maximum long term depression factor of the parallel fibers-Purkinje cells connection [2],

$$ltd\_PF\_PC\_max = \left[\left[ltd^{pf-pc}_{uml_1,ccm_1}, ..., ltd^{pf-pc}_{uml_1,ccm_M}\right], ..., \left[ltd^{pf-pc}_{uml_N,ccm_1}, ..., ltd^{pf-pc}_{uml_N,ccm_M}\right]\right].$$

alpha_PC_DCN (*float 2-dimensional array, n_uml × n_ccm*, Default = 1.)

Decaying factor of the Purkinje cells-deep cerebellar nuclei connection [2],

$$alpha\_PC\_DCN = \left[\left[\alpha^{pc-dcn}_{uml_1,ccm_1}, ..., \alpha^{pc-dcn}_{uml_1,ccm_M}\right], ..., \left[\alpha^{pc-dcn}_{uml_N,ccm_1}, ..., \alpha^{pc-dcn}_{uml_N,ccm_M}\right]\right].$$

ltp_PC_DCN_max (*float 2-dimensional array, n_uml × n_ccm*, Default = $10.^{-3}$)

Maximum long term potentiation factor of the Purkinje cells-deep cerebellar nuclei connection [2],

$$ltp\_PC\_DCN\_max = \left[\left[ltp^{pc-dcn}_{uml_1,ccm_1}, ..., ltp^{pc-dcn}_{uml_1,ccm_M}\right], ..., \left[ltp^{pc-dcn}_{uml_N,ccm_1}, ..., ltp^{pc-dcn}_{uml_N,ccm_M}\right]\right].$$

ltd_PC_DCN_max (*float 2-dimensional array, n_uml × n_ccm*, Default = $10.^{-3}$)

Maximum long term depression factor of the Purkinje cells-deep cerebellar nuclei connection [2],

$$ltd\_PC\_DCN\_max = \left[\left[ltd^{pc-dcn}_{uml_1,ccm_1}, ..., ltd^{pc-dcn}_{uml_1,ccm_M}\right], ..., \left[ltd^{pc-dcn}_{uml_N,ccm_1}, ..., ltd^{pc-dcn}_{uml_N,ccm_M}\right]\right].$$

alpha_MF_DCN (*float 2-dimensional array*, *n_uml* × *n_ccm*, Default = 1.)
Decaying factor of the mossy fibers-deep cerebellar nuclei connection [2],

$$alpha\_MF\_DCN = \left[\left[\alpha^{mf-dcn}_{uml_1,ccm_1}, ..., \alpha^{mf-dcn}_{uml_1,ccm_M}\right], ..., \left[\alpha^{mf-dcn}_{uml_N,ccm_1}, ..., \alpha^{mf-dcn}_{uml_N,ccm_M}\right]\right].$$

ltp_MF_DCN_max (*float 2-dimensional array*, *n_uml* × *n_ccm*, Default = $10.^{-3}$)
Maximum long term potentiation factor of the mossy fibers-deep cerebellar nuclei connection [2],

$$ltp\_MF\_DCN\_max = \left[\left[ltp^{mf-dcn}_{uml_1,ccm_1}, ..., ltp^{mf-dcn}_{uml_1,ccm_M}\right], ..., \left[ltp^{mf-dcn}_{uml_N,ccm_1}, ..., ltp^{mf-dcn}_{uml_N,ccm_M}\right]\right].$$

ltd_MF_DCN_max (*float 2-dimensional array*, *n_uml* × *n_ccm*, Default = $10.^{-3}$)
Maximum long term depression factor of the mossy fibers-deep cerebellar nuclei connection [2],

$$ltd\_MF\_DCN\_max = \left[\left[ltd^{mf-dcn}_{uml_1,ccm_1}, ..., ltd^{mf-dcn}_{uml_1,ccm_M}\right], ..., \left[ltd^{mf-dcn}_{uml_N,ccm_1}, ..., ltd^{mf-dcn}_{uml_N,ccm_M}\right]\right].$$

alpha_IO_DCN (*float 2-dimensional array*, *n_uml* × *n_ccm*, Default = 1.)
Decaying factor of the inferior olive-deep cerebellar nuclei connection [18],

$$alpha\_IO\_DCN = \left[\left[\alpha^{io-dcn}_{uml_1,ccm_1}, ..., \alpha^{io-dcn}_{uml_1,ccm_M}\right], ..., \left[\alpha^{io-dcn}_{uml_N,ccm_1}, ..., \alpha^{io-dcn}_{uml_N,ccm_M}\right]\right].$$

mtp_IO_DCN_max (*float 2-dimensional array*, *n_uml* × *n_ccm*, Default = $10.^{-3}$)
Potentiation modulating term of the inferior olive-deep cerebellar nuclei connection [18],

$$mtp\_IO\_DCN\_max = \left[\left[mtp^{io-dcn}_{uml_1,ccm_1}, ..., mtp^{io-dcn}_{uml_1,ccm_M}\right], ..., \left[mtp^{io-dcn}_{uml_N,ccm_1}, ..., mtp^{io-dcn}_{uml_N,ccm_M}\right]\right].$$

mtd_IO_DCN_max (*float 2-dimensional array*, *n_uml* × *n_ccm*, Default = $10.^{-3}$)
Depression modulating term of the inferior olive-deep cerebellar nuclei connection [18],

$$mtd\_IO\_DCN\_max = \left[\left[mtd^{io-dcn}_{uml_1,ccm_1}, ..., mtd^{io-dcn}_{uml_1,ccm_M}\right], ..., \left[mtd^{io-dcn}_{uml_N,ccm_1}, ..., mtd^{io-dcn}_{uml_N,ccm_M}\right]\right].$$

2.5 Functions

create_model()

Initialize the LWPR with the parameters defined in section 2.4.1. The function have to be call after the initialization of the cerebellum. Call the class object ***model*** to access directly to the LWPR parameters.

update_model()

Update the LWPR model given the x_MF and the x_MF_teach (Section 2.3).

prediction(error)

This function output the deep cerebellar nuclei contribution given the input error signal,

$error_{n\_ccm \times n\_uml} = \left[\mathbf{e}_{ccm_1}, ..., \mathbf{e}_{ccm_N}\right] = \left[\left[e_{uml_1,ccm_1}, ..., e_{uml_N,ccm_1}\right], ..., \left[e_{uml_1,ccm_N}, ..., e_{uml_N,ccm_N}\right]\right];$

3 Example Script File

In this section, an example of usage is presented. The script intends to use the class to predict the next control signal and add adaptive correction. Before starting, it is important to make sure your computer has python2.7 and the LWPR Locally Weighted Projection Regression packages installed. To install the class run the '*__init__.py*' script first.

```
Define the signals range for both IO and DCN
/* Define all the necessary variable */
n_uml = 3
n_ccm = 1
n_input_mf = 9

/* Initialize the cerebellum class */
cerebellum = MCC(n_uml, n_ccm, n_input_mf)

/* Override variables */
cerebellum.name = "test model"
cerebellum.range_IO = io_range
cerebellum.range_signal = signal_range
cerebellum.init_D_gr = 10.

/* Initialize the LWPR */
cerebellum.create_model()

for $k = 0$, samples do
    /* Get the current sensory signal $x^{mf}(k)$ */
    $cerebellum.x\_MF = [x_1^{mf}(k), \dots , x_{n\_input\_mf}^{mf}(k)]$
    /* Get the latest signal $x^{tot}(k-1)$ */
    $cerebellum.x\_MF\_teach = [x_1^{tot}(k-1), \dots , x_{n\_uml}^{tot}(k-1)]$
    /* Update the LWPR */
    cerebellum.update_model()

    /* Compute the error given the current state information */
    /* Get the cerebellar contribution */
    $\Delta \mathbf{x}^{dcn}$ = cerebellum.prediction(error)

end
```

Algorithm 1: Cerebellum class example script

References

[1] N. R. Luque, J. A. Garrido, R. R. Carrillo, E. D'Angelo, and E. Ros, "Fast convergence of learning requires plasticity between inferior olive and deep cerebellar nuclei in a manipulation task: A closed-loop robotic simulation", *Frontiers in computational neuroscience*, vol. 8, p. 97, 2014.

[2] J. A. Garrido Alcazar, N. R. Luque, E. D'Angelo, and E. Ros, "Distributed cerebellar plasticity implements adaptable gain control in a manipulation task: A closed-loop robotic simulation", *Frontiers in neural circuits*, vol. 7, p. 159, 2013.

[3] M. C. Capolei, N. A. Andersen, H. H. Lund, E. Falotico, and S. Tolu, "A cerebellar internal models control architecture for online sensorimotor adaptation of a humanoid robot acting in a dynamic environment", *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 80–87, 2019.

[4] M. C. Capolei, E. Angelidis, E. Falotico, H. H. Lund, and S. Tolu, "A biomimetic control method increases the adaptability of a humanoid robot acting in a dynamic environment", *Frontiers in Neurorobotics*, vol. 13, p. 70, 2019, ISSN: 1662-5218. DOI: 10.3389/fnbot.2019.00070. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2019.00070.

[5] M. Ito, *The cerebellum and neural control.* Raven press, 1984.

[6] T. J. Ebner, A. L. Hewitt, and L. S. Popa, "What features of limb movements are encoded in the discharge of cerebellar neurons?", *The Cerebellum*, vol. 10, no. 4, pp. 683–693, 2011.

[7] S. Kitazawa, T. Kimura, and P.-B. Yin, "Cerebellar complex spikes encode both destinations and errors in arm movements", *Nature*, vol. 392, no. 6675, p. 494, 1998.

[8] C. I. De Zeeuw, C. C. Hoogenraad, S. Koekkoek, T. J. Ruigrok, N. Galjart, and J. I. Simpson, "Microcircuitry and function of the inferior olive", *Trends in neurosciences*, vol. 21, no. 9, pp. 391–400, 1998.

[9] D. Marr, "A theory of cerebellar cortex", *The Journal of physiology*, vol. 202, no. 2, pp. 437–470, 1969.

[10] E. D'Angelo, A. Antonietti, S. Casali, C. Casellato, J. A. Garrido, N. R. Luque, L. Mapelli, S. Masoli, A. Pedrocchi, F. Prestori, M. F. Rizza, and E. Ros, "Modeling the cerebellar microcircuit: New strategies for a long-standing issue", *Frontiers in Cellular Neuroscience*, vol. 10, p. 176, 2016, ISSN: 1662-5102. DOI: 10.3389/fncel.2016.00176.

[11] S. Tolu, M. Vanegas, N. R. Luque, J. A. Garrido, and E. Ros, "Bio-inspired adaptive feedback error learning architecture for motor control", *Biological Cybernetics*, vol. 106, no. 8-9, pp. 507–522, 2012, ISSN: 0340-1200.

[12] S. Tolu, M. Vanegas, J. A. Garrido, N. R. Luque, and E. Ros, "Adaptive and predictive control of a simulated robot arm", *Int. J. Neural Syst.*, vol. 23, no. 3, 2013.

[13] J. S. Albus, "A theory of cerebellar function", *Mathematical Biosciences*, vol. 10, no. 1-2, pp. 25–61, 1971.

[14] S. Vijayakumar and S. Schaal, “Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space”, in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, vol. 1, 2000, pp. 288–293.

[15] E. Todorov and M. I. Jordan, “Optimal feedback control as a theory of motor coordination”, *Nature neuroscience*, vol. 5, no. 11, p. 1226, 2002.

[16] S. H. Scott, “The computational and neural basis of voluntary motor control and planning”, *Trends in Cognitive Sciences*, vol. 16, no. 11, pp. 541 –549, 2012.

[17] K. Doya, H. Kimura, and M. Kawato, “Neural mechanisms of learning and control”, *IEEE Control Systems Magazine*, vol. 21, no. 4, pp. 42–54, 2001.

[18] N. R. Luque, R. R. Carrillo, F. Naveros, J. A. Garrido, and M. Sáez-Lara, “Integrated neural and robotic simulations. simulation of cerebellar neurobiological substrate for an object-oriented dynamic model abstraction process”, *Robotics and Autonomous Systems*, vol. 62, no. 12, pp. 1702 –1716, 2014.