HOW CEREBELLAR ARCHITECTURE FACILITATES RAPID ONLINE LEARNING

Adriana Perez Rotondo[1,*], **Dhruva Venkita Raman** [1,2], **and Timothy O'Leary** [1]

[1]Department of Engineering, University of Cambridge, UK
[2]Department of Informatics, University of Sussex, UK
[*]ap2013@cam.ac.uk

ABSTRACT

The cerebellum has a distinctive circuit architecture comprising the majority of neurons in the brain. Marr-Albus theory and more recent extensions [1, 40, 38, 15] demonstrate the utility of this architecture for particular types of learning tasks related to the separation of input patterns. However, it is unclear how the circuit architecture facilitates known functional roles of the cerebellum. In particular, the cerebellum is critically involved in refining motor plans even during the ongoing execution of the associated movement. Why would a cerebellar-like circuit architecture be effective at this type of 'online' learning problem? We build a mathematical theory, reinforced with computer simulations, that captures some of the particular difficulties associated with online learning tasks. For instance, synaptic plasticity responsible for learning during a movement only has access to a narrow time window of recent movement errors, whereas it ideally depends upon the entire trajectory of errors, from the movement's start to its finish. The theory then demonstrates how the distinctive input expansion in the cerebellum, where mossy fibre signals are recoded in a much larger number of granule cells, mitigates the impact of such difficulties. As such, the energy cost of this large, seemingly redundantly connected circuit might be an inevitable cost of precise, fast, motor learning.

***K*eywords** Cerebellum · Motor control · Motor learning · Network architecture

1 Introduction

The cerebellar cortex has a very distinct circuit architecture characterised by a large 'input expansion': each mossy fibre input typically projects to 250 granule cells, a population that comprises more than half of the neurons in the brain [21]. Why? This question has spurred a long line of scientific studies.

A notable proposed justification for this architecture comes in the form of Marr-Albus theory. Marr, Albus, and more recent scientists, have shown how this expansion facilitates 'pattern separation' [15, 16, 8]. That is, large numbers of highly overlapping cerebellar inputs corresponding to distinct but correlated stimuli, can be accurately resolved into less-correlated patterns of cerebellar output, facilitating distinct behavioural responses. Marr Albus theory remains highly influential, despite its age [57].

However, there is a gap between the proposed functional role of pattern separation, and the actual functional roles that the cerebellum is known to be involved in. We know that cerebellar involvement is necessary for the precise execution and real-time calibration of movements [62, 17], but not if or how a pattern separation module is useful for this [33].

We bridge this gap by building an alternative theoretical account that justifies how the cerebellar input expansion facilitates online motor learning. We start by considering an influential model of motor control first proposed by Kawato et al. [31], although our results are not specific to the model. Here, the cerebral sensorimotor cortex produces an approximate motor command, which is refined by cerebellar input. Cerebellar processing of motor commands is continually adapted based upon ongoing sensation of motor errors. Overall, the system allows motor commands to adapt on a fast timescale to new contexts that alter biomechanics (e.g. a sprained ankle or a carrying an unusual object) that would otherwise result in motor errors. Conceptually, the cerebellum, in this setup, maintains a constantly updating *inverse model* of the musculoskeletal system: a prediction of which motor commands are required to effect a particular

somatic movement [73]. This inverse model allows the cerebellum to refine approximate motor commands sent by other brain areas to produce a desired movement [17, 5, 7, 47, 52, 63, 41].

There are limits to the efficacy of cerebellar adaptation in reducing motor errors using plausible plasticity rules, both within this model and in general. One issue is that perfect adaptation would require information on the entire trajectory of movement errors, spanning the start and end of a movement. Biologically, information on movement errors seems to be restricted to a narrow time-window of the recent past, as only these movement errors are known to be within the receptive field of climbing fibres, whose complex spikes signal movement error and induce adaptation. Another is that information available to the cerebellum as a whole may not be available to each and every plasticising synapse, due to biophysical constraints on the speed and completeness of communication between synapses. We refer to erroneous adaptations arising from these two sources as 'information errors' and 'learning rule errors' respectively.

In this work, we mathematically formalise the effect of information errors and learning rule errors on the ability of plasticity rules in the cerebellum to accurately update inverse models of the musculoskeletal system. We analytically quantify how this effect changes within a cerebellar model as extra granule cells are introduced. We show that this effect reduces, in such a way that the propensities to learn fast and accurately are simultaneously increased, agnostic to the particular set of movements being learned. Our analytic work is validated by numerical simulation using the model of [33]. Overall, our work suggests that the distinctive, energetically expensive, 'input expansion' feature of the cerebellar architecture may represent an inevitable cost of fast, precise, motor learning.

2 Results

2.1 A standard model of cerebellar involvement in motor control

We consider a well studied model of motor control in which the motor cortex and cerebellum combine to produce precise motor commands that can adapt to changing circumstances such as performing new movements or compensating for musculoskeletal changes from growth or injury (see Figure 1 A) [31, 32]. In this model, part of the motor system, including the motor cortex, transforms upstream neural signals, corresponding to desired somatic movements, into motor commands. The motor system can modify its motor commands by taking into account sensory information on the mismatch between the desired and actual somatic positions and movements, using negative feedback (see Methods 4.1).

Negative feedback cannot perfectly correct for mismatches between a planned and desired movement. Biophysical delays between a movement mismatch, its sensation, and its correction can destabilise the movement [11, 43, 44, 58]. As an example, imagine the overbalancing of a novice balancing on a tightrope, and correcting for leftwards sway with delayed rightwards movement. Indeed, the uncontrolled oscillation is reminiscent of the movements of a patient with cerebellar damage. In this model of motor control, the cerebellum refines motor commands by supplementing the aforementioned reactive correction (negative feedback) with a learned, predictive correction. Imagine the tightrope walker, with practice, reacting to leftwards sway with a more nuanced correction that takes into account their future, desired, movement trajectory.

Concretely, we model the cerebellum as receiving information on a movement plan extending into the future, through its mossy fibre inputs. The neural representation of the planned movement at time t' is denoted $r(t')$, and the plan must extend into the future by some timespan Δt_I and minimally include information on intermediate timepoints with some spacing δt_I. Thus, the vector of all the I mossy fibre firing rates at time t is given (see Figure 1A) as

$$\mathbf{m}(t) = [r(t), r(t+\delta t_I), \ldots, r(t+\Delta t_I)]^T. \tag{1}$$

The cerebellum must then transform its mossy fibre input into a (possibly imperfect) modification of the motor command generated by the motor cortex. The input is first recoded in the cerebellar granule layer, consisting of N granule cells. We model the firing rate $h_i(t)$ of the ith granule cell as

$$h_i(t) = \sigma\left(\sum_{j=1}^{I} z_{ij}(t) m_j(t) + b_i(t)\right), \tag{2}$$

where σ is the nonlinear activation function, and $z_{ij}(t)$ is the synaptic weight strength from the j'th input mossy fibre to the i'th granule cell at time t. The actual firing rate of a granule cell, given a degree of synaptic input from the mossy fibres, depends upon unmodelled factors such as inhibition from basket and stellate cells. Such factors are lumped in the *bias* term $b_i(t)$, which decreases with increasing constitutive inhibition. Granule cell activity then determines the firing rate of a single output Purkinje cell:

$$p(t) = \sum_{i=1}^{N} w_i(t) h_i(t) := \mathbf{w}^T(t)\mathbf{h}(t), \tag{3}$$

where $\mathbf{w}(t)$ is the vector of synaptic weights connecting the purkinje cell to granule cells through the parallel fibres.

Plasticity in the parallel fibre synaptic weights $\mathbf{w}(t)$ arises from real-time mismatch between the planned movement and sensory information on the actual movement, and acts to reduce mismatch on future instantiations of the movement [23, 24, 34, 66]. Biologically, this plasticity is induced by complex spikes in the climbing fibres, which activate in a narrow time window (50-200ms) following movement errors. As such, we model the error signal available to induce plasticity as the aggregate mismatch over a narrow time window Δt_e that terminates by an amount Δt_r in the past, reflecting biophysical delays between movement error and consequent plasticity:

$$e(t) = \frac{1}{\Delta t_e} \int_{t-\Delta t_e-\Delta t_r}^{t-\Delta t_r} y(t', \mathbf{w}) - r(t') dt'. \tag{4}$$

We don't know the exact plasticity rules that transmute this error signal into cerebellar adaptation. Instead, we assume that biologically plausible learning rules equate to some form of imperfect gradient descent, regardless of mechanism [53]. As such we can model plasticity as a combination of gradient descent and a tunable degree of noise:

$$\delta \mathbf{w}_t = -\gamma e(t) \mathbf{h}(t - \Delta t_h) + \eta \hat{\epsilon} \tag{5}$$

where $\hat{\cdot}$ denotes a normalised vector, $0 < \gamma \ll 1$ is the degree of plasticity in the gradient descent direction, Δt_h is the delay between purkinje cell input and climbing fibre signal that leads to synaptic plasticity. Imperfections are modelled through a vector ϵ, which is componentwise Gaussian. $\eta\hat{\epsilon}$ is a multiplier determining the degree of this noise term. The effective signal-to-noise ratio is then $\frac{\gamma}{\eta}$, which is kept constant. Note that the gradient portion of this learning rule equates to the well-known LMS rule [72], and is local in the sense that each synapse need only respond to its peri-synaptic firing rates.

2.2 Motivating example

We now simulate the summarised model, showing how ongoing cerebellar plasticity iteratively refines the motor commands underlying a repeatedly practiced movement, thus increasing movement precision over time. We will also highlight changes in the speed and precision of learning arising from changes in the number of granule cells.

We consider an arbitrary movement performed over a timespan $[0, T]$. Synaptic weights in the cerebellar module, which are randomly initialised (see Methods 4.1), update according to the learning rule of (5). The dynamics of synaptic plasticity and the musculoskeletal system are both continuous; the cerebellar weights are changing while the movement is being actuated. However, these two dynamics must have different timescales. Learning must be slower than the dynamics of the system to avoid interference. Computer simulation requires that updates are temporally discretised with some small period δt_e. For details of the simulations see Methods 4.1.

Initially, only negative feedback modulating motor commands from the motor cortex ties the actual movement to the desired movement, resulting in high degrees of mismatch. Over time, cerebellar learning decreases the mismatch, by effectively learning a predictive model of appropriate motor command modulations in response to sensed mismatch. This is visible in Figure 1.

How could one increase the effectiveness of a learning rule? Naively, one could increase the plasticity induced by a given error signal to speed learning (increasing γ in 5). We refer to this as increasing the *gain* of the learning rule. An excessive gain can, however, slow or destabilise learning: imperfections in the plasticity direction are amplified. So at any point there is an optimal gain, which decreases with the level of learning. We hypothesise that the optimal gain is impossible to accurately compute. It requires information on the second order terms at each time step [54].

This implies the existence of a trade-off between learning speed and steady state loss. Learning speed quantifies how fast the loss can be reduced at initial parts of learning and steady state loss how good the performance is at the end of learning (see Figure 1 C) and Methods 4.3). Excessive learning gain speeds initial learning, at the cost of steady state precision. Insufficient gain has the opposite effect (see Figure 2 B)).

In Figure 2, we iteratively increase the number of granule cells, while keeping all other aspects of the learning task constant. Each new granule cell forms input connections with $K = 4$ randomly chosen input mossy fibres, and an output connection with the single purkinje cell (see Methods 4.2). Remarkably, both the learning speed and the steady state loss improve, although this improvement saturates (see Figure 2 C) and D)). Thus, on our minimal example, 'excessive' numbers of granule cells that are not strictly required for perfect task learning, nevertheless aid learning. Is this property generic? Why does it happen?

2.3 Sources of error in online motor learning

To answer the questions raised in the previous section, we need to consider exactly what makes the cerebellum's job of online motor learning difficult in our model. We start by introducing a performance metric for the entire movement,

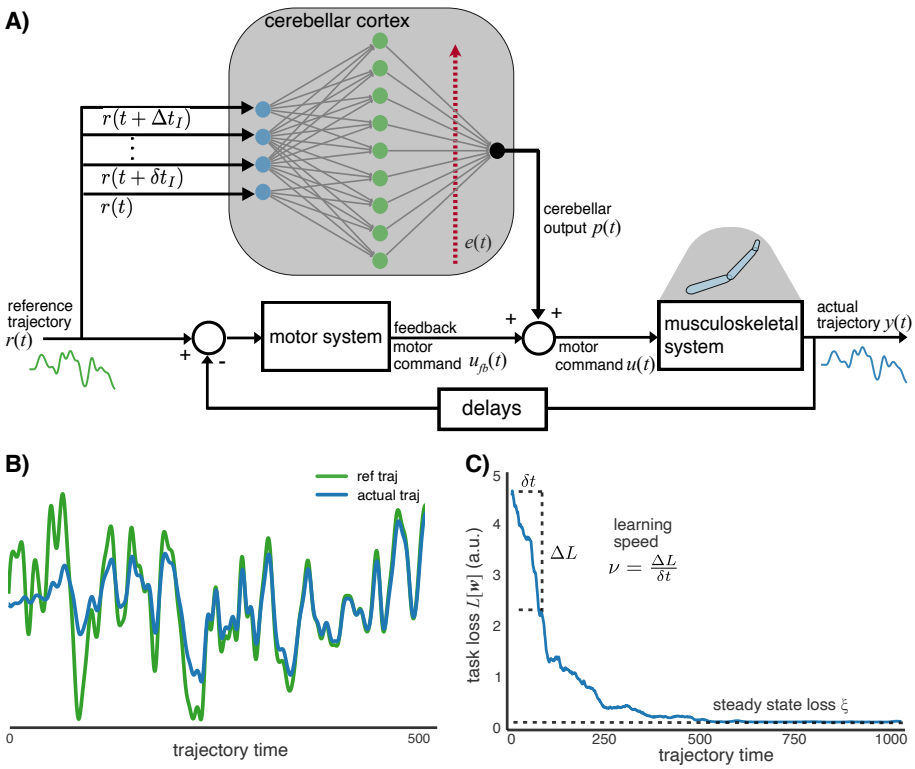

Figure 1: System used to model the cerebellum for motor control and learning. **A)** Diagram of the modeled system for the trajectory tracking task. The controlled object is the musculoskeletal system that needs to produce the reference trajectory. The feedback controller represents the part of the motor system which generates a motor command that is modulated by the cerebellar network. The cerebellar network receives as input a motor plan and outputs a motor command. The sum of the two motor commands, generated by the cerebellar network and motor cortex constitutes the total command sent to the controlled object. **B)** The cerebellar cortex network learns online during the movement by modifying the output weights according to the error information carried by the climbing fibres. We choose an LMS inspired learning rule given by eq (5). Initially the actual trajectory is far form the reference trajectory. As learning progresses, the two curves come closer together. **C)** The task loss $L[\mathbf{w}]$ defined in eq (6) quantifies the error between the reference and actual trajectories over the whole movement if the weights were fixed to some value $\mathbf{w}$. We calculate the task loss after each weight update. The task loss decreases throughout the movement until it reaches a steady state value referred as the *steady state loss*. The *learning speed* quantifies how fast the task loss decreases initially.

which we call the *task loss*, and which depends upon the cerebellar synaptic weights:

$$L[\mathbf{w}] = \frac{1}{2T}\int_{t=0}^{T} ||y(t;\mathbf{w}) - r(t)||_2^2 dt. \tag{6}$$

This is the performance metric that synaptic plasticity should act to minimise. However, evaluating the task loss at a point in time is unlikely as

1. it requires sensing of the entire movement trajectory.
2. any online synaptic plasticity rule, such as equation (5), precludes the weights from remaining constant over the entire movement trajectory.

Learning online, while performing the movement itself, introduces a narrow time window that severely limits the information available for synaptic plasticity mechanisms to appropriately adjust synaptic weights. As mentioned previously, we expect the climbing fibres to signal an analogue of the online error (equation (4)) to induce synaptic plasticity. As such, we assume that the performance metric that the cerebellum plausibly has access to is the squared magnitude of the online error, which we call the *online loss*:

$$L^o[t;\mathbf{w}] = \frac{1}{2(\Delta t_e)}\int_{t'=t-\Delta t_e-\Delta t_r}^{t-\Delta t_r} ||y(t';\mathbf{w}) - r(t')||_2^2 dt'. \tag{7}$$

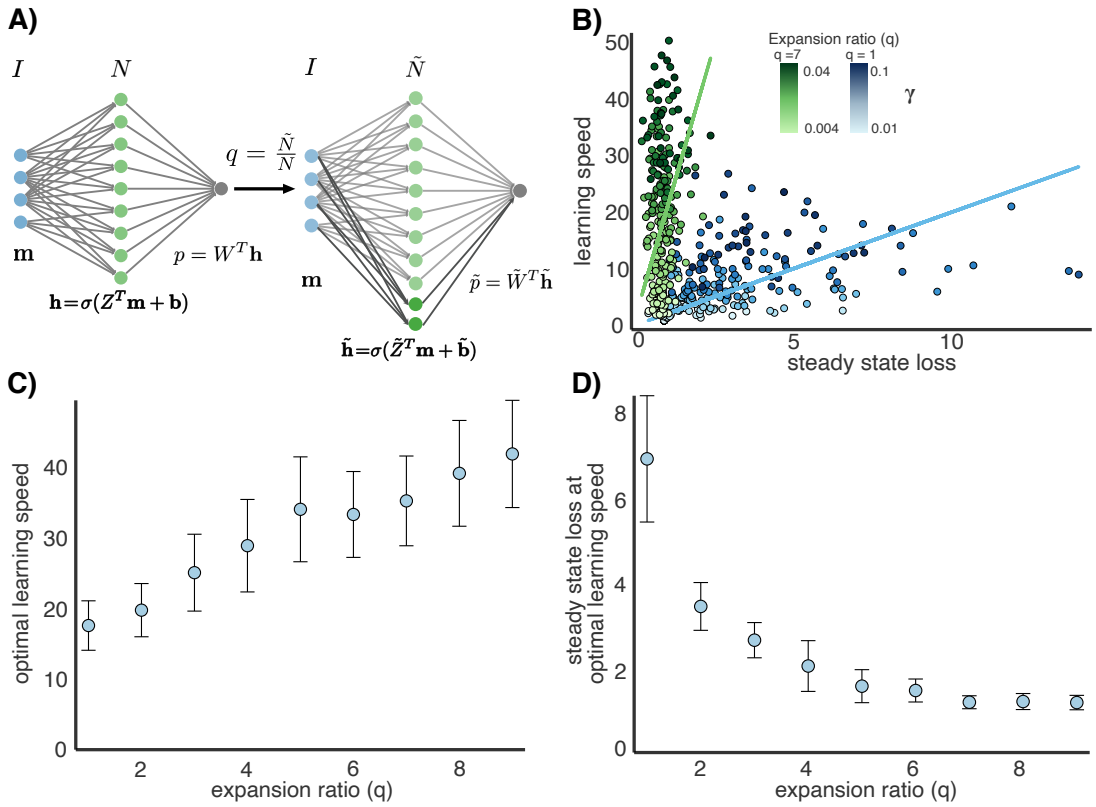

Figure 2: Granule cell layer expansion increases learning performance. **A)** We consider a network expansion that adds granule cells to the hidden layer. The expansion maintains the number of input mossy fibres but increases the number of granule cells from N to $\tilde{N}$. The expansion ratio is defined as $q = \frac{\tilde{N}}{N}$. The network expansion changes the granule cell layer activity. **B)** Adding granule cells navigates the trade-off between the learning speed and steady state loss. We plot the trade-off for two different expansion ratios: $q = 1$ in blue, $q = 7$ in green. The shade of the color denotes the value of γ. The networks with more granule cells can achieve better learning speed for some steady state loss. **C)** We compute the optimal learning speed over all γ for each network expansion. The optimal learning speed increases with network expansion. **D)** We plot the steady state loss for the γ that gives optimal learning speed (value in **C**)). The steady state loss decreases with network size indicating that the larger network can achieve better learning speed to steady state loss trade-off. Error bars indicate ± 1 SEM over simulations with different reference trajectories and weight initilisations. For details on the methods to generate this graph see 4.3.

,where Δt_e is the online loss time window, and Δt_r is the online loss delay time and $y(t, \mathbf{w})$ is the actual trajectory produced by the musculoskeletal system. Δt_e is determined by how much memory of the error the motor system has access to and Δt_r is determined by reaction times, and delays in the error signals carried by the climbing fibres to the cerebellar cortex [67, 42, 65]. Note that integrating the online loss over an excessive time window Δt_e is not necessarily desirable. The cerebellar synaptic weights are constantly changing, and as such online errors produced further in the past are less indicative of online errors that would be produced by the current system in the same setting.

A synaptic plasticity rule looking to decrease the online loss L^o may not effectively decrease the overall task loss L. We can quantify this. In general, the local direction of synaptic plasticity that maximally changes a loss function L is known as the gradient of L, and is denoted ∇L. A 'perfect' learning rule would satisfy

$$\delta \mathbf{w}_t \propto -\nabla L[\mathbf{w}] \tag{8}$$

Any other learning rule would decrease the loss function by a smaller amount, for any given overall degree of synaptic plasticity.

A cerebellar learning rule only has access to the online loss, hence, in the best case scenario such a rule would perfectly reduce the online loss L^o. In the sense of equation (8), we would have $\delta \mathbf{w}_t \propto -\nabla L^o[\mathbf{w}_t]$. Realistically, ∇L^o might be

hard to compute and induce perfectly. As such, we model a generic, imperfect, cerebellar learning rule as

$$\delta \mathbf{w}_t = -\gamma_t \nabla \hat{L}_t^o + \eta_t^{lr} \hat{\epsilon}_t^{lr}. \tag{9}$$

where $\hat{\cdot}$ denotes a normalised vector, $\nabla L_t^o \equiv \nabla_{\mathbf{w}} L^o[t; \mathbf{w}_t]$ the online gradient. Here $\hat{\epsilon}_t^{lr}$ is a normalised vector of Gaussian random variables, representing generic imperfections in the learning rule. η^{lr} is a tunable scalar representing the strength of these imperfections, and we refer collectively to $\eta^{lr}\hat{\epsilon}_t^{lr}$ as the *learning rule error*. γ_t quantifies the proportion of a weight change in the direction of the online gradient; it quantifies the sensitivity of the learning system to error signal. We will refer to learning rules acting on the online error, and belonging to the family specified by equation (9), as *online learning rules*.

To what extent would an online learning rule effectively reduce the task loss? To answer this question, we can rewrite equation (9) in terms of the task loss. First note that

$$\nabla \hat{L}_t^o = \gamma_t^o \nabla \hat{L}_t + \eta_t^o \hat{\epsilon}_t^o. \tag{10}$$

where $\nabla L_t \equiv \nabla_{\mathbf{w}} L[\mathbf{w}_t]$ is the task gradient. Here, $\hat{\epsilon}_t^o$ represents the direction of discrepancy between the online and task gradients, and η^o the magnitude. Together, $\eta_t^o \hat{\epsilon}_t^o$ is referred to as the *online error*. Thus, we can rewrite equation (9) as

$$\delta \mathbf{w}_t = -\gamma_t (\gamma_t^o \nabla \hat{L}_t + \eta_t^o \hat{\epsilon}_t^o) + \eta_t^{lr} \hat{\epsilon}_t^{lr} \tag{11}$$

Fig 3 B) illustrates the decomposition of the change in weights.

There are two circumstances in which an online learning rule would learn as well as one directly sensing the task loss. One is where the gradients of the online and task losses correspond, and thus $\eta^o = 0$. Another is where $\eta^o \neq 0$, but the direction of plasticity induced by the online error is benign: it doesn't compromise improvements in task loss. More generally, we need to consider both the degree and destructiveness of the online error component in a learning rule.

Intuitively, for an imperfect learning rule, doubling the degree of plasticity would double both the degree of 'good' plasticity (in the direction of the task gradient) and 'erroneous' plasticity (from online and learning rule errors): the signal-to-noise ratio shouldn't change. Hence, we define the *quality* of an online learning rule as

$$\sigma_t := \frac{\gamma_t}{\eta_t^{lr}}, \tag{12}$$

and note that it is independent of the magnitude of plasticity in response to a given error signal. Indeed, we could rewrite equation (9) as $\delta \mathbf{w}_t = -\gamma_t (\nabla L_t^o + \frac{1}{\sigma_t} \hat{\epsilon}_t^{lr})$. σ_t quantifies the quality of the learning rule determined by the biophysical systems regulating synaptic plasticity.

2.4 The geometry of the loss landscape controls learning performance

In the previous section we showed that learning online with a biologically plausible learning rule introduces two types of error terms for learning: the *online learning error* and the *learning rule error*. These two errors capture the two sources of limitation for learning: the online learning error arises from having limited information on the task loss to determine the optimal plasticity direction, and the learning rule error captures the difficulty to produce synaptic plasticity in that optimal synaptic plasticity direction using a local learning rule.

Consider a change in weights $\delta \mathbf{w}_t$ over the interval δt. Synaptic plasticity is a continuous process, however, for our analysis we have to discretise it. We assume that plasticity is in the same direction over the interval δt. Consider a $\delta \mathbf{w}_t$ given by (11), it leads to a change in task loss

$$\Delta L_t = L[\mathbf{w}_t + \delta \mathbf{w}_t] - L[\mathbf{w}_t] \tag{13}$$

If the weight update is beneficial for learning, then ΔL_t is negative. The change in task loss ΔL_t is determined by the geometry of the loss landscape (i.e. the relation between the task loss and the weights) [54, 55]. Indeed, using the Taylor expansion, we can write the change in task loss

$$\Delta L_t = \delta \mathbf{w}_t{}^T \nabla L_t + \frac{1}{2} \delta \mathbf{w}_t{}^T \mathbf{H}_t \delta \mathbf{w}_t + O(||\delta \mathbf{w}_t||^3) \tag{14}$$

where $\mathbf{H}_t = \nabla^2 L[\mathbf{w}_t]$ is the hessian of the task loss and O refers to the big O notation. The first term $\delta \mathbf{w}_t{}^T \nabla L_t$ quantifies the effect of the slope $||\nabla L_t||$ of the loss landscape on learning. For learning to occur, the synaptic change must anti-correlate with the task gradient $\delta \hat{\mathbf{w}}_t{}^T \nabla \hat{L}_t < 0$ The larger the slope, the larger change in task loss. The second term quantifies the effect of the curvature of the loss landscape. If $\delta \mathbf{w}_t$ projects onto directions of positive (upward) curvature, the second term will be positive and slow down learning (make ΔL_t less negative).

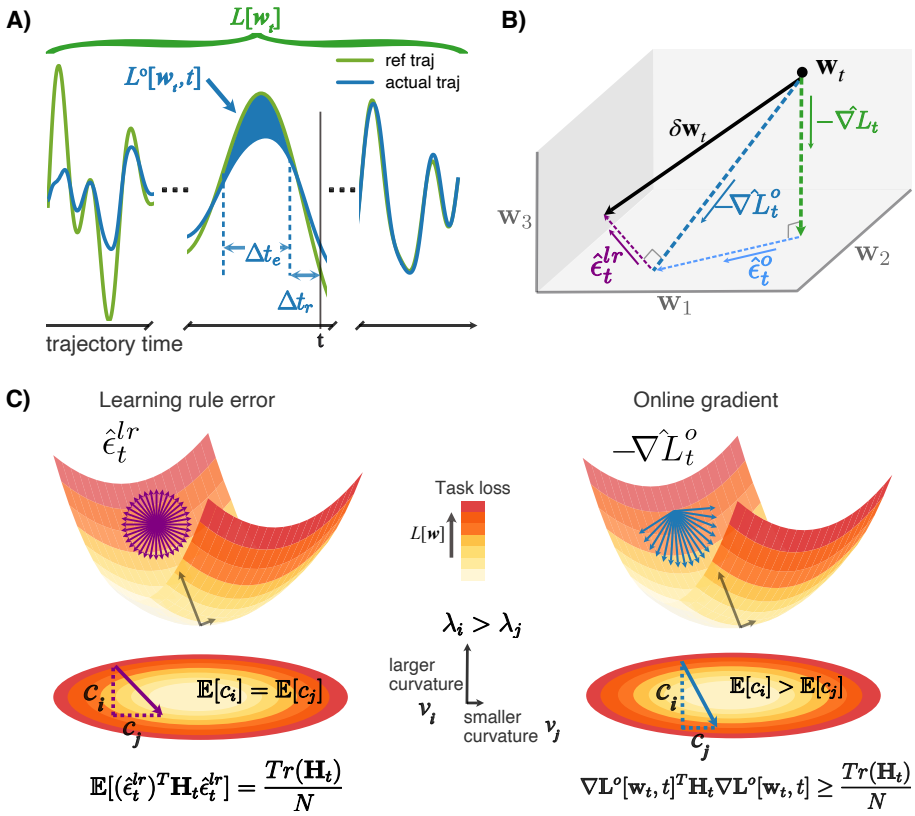

Figure 3: Online learning limits the amount of error information accessible for synaptic plasticity mechanisms to appropriately adjust synaptic weights. **A)** The task loss $L[\mathbf{w}]$ measures the distance between the reference and actual trajectories over the whole movement. At some time t during the movement (black vertical dotted line), the system only has access to the online loss $L^o[\mathbf{w}_t, t]$, the distance between the two trajectories over a small window of time in the past (blue shaded region). **B)** The task loss gradient $-\nabla\hat{L}_t$ and the online loss gradient $-\nabla\hat{L}_t^o$ might not be in the same direction in weight space. We refer to the difference between the two as the online error direction $\hat{\epsilon}_t^o$. The difference between the direction of change in weights $\delta\hat{\mathbf{w}}_t$ and $-\nabla\hat{L}_t^o$ is called the learning rule error direction $\hat{\epsilon}_t^{lr}$. The change in weights can be decomposed as in (11). **C)** Depiction of the projection of the learning rule error (on the left) and the online gradient (on the right) onto the hessian of the task loss. We use a simplified loss function of two weights. On the top, representation of the loss landscape where the z-axis corresponds to the task loss $L[\mathbf{w}]$. Every point on the landscape corresponds to a different network state $\mathbf{w}$ with task loss $L[\mathbf{w}]$. On the bottom the bird-eye view of the loss landscape. The color represents the value of the task loss. Near a local minimum the loss landscape can be approximated by a quadratic function with constant hessian with two eigenvectors. $\mathbf{v}_i$ has a large eigenvalue λ_i and $\mathbf{v}_j$ has smaller eigenvalue λ_j on the y and x directions respectively. The learning rule error $\hat{\epsilon}_t^{lr}$ is uncorrelated with the task loss. It can be modelled as a random vector that could be in any direction in weight space (top diagram). On expectation, $\hat{\epsilon}_t^{lr}$ projects evenly onto all the eigenvectors of the hessian (bottom diagram). Hence, on expectation its projection onto the hessian is equal to the average curvature. The online gradient, is not independent of the task loss. It projects more strongly onto directions of larger curvature. Hence its projection onto the hessian will be larger than the average curvature.

How is the particular change in synaptic weights given by (11) affected by the geometry of the loss landscape? For the first term, we find

$$\mathbb{E}[\delta\mathbf{w}_t{}^T\nabla L_t] = -\gamma_t\gamma_t^o||\nabla L_t|| \tag{15}$$

where $\mathbb{E}$ denotes the expectation over all the possible directions of $\hat{\epsilon}_t^{lr}$. We have used the fact that the online error $\hat{\epsilon}_t^o$ is perpendicular to the task gradient ∇L_t and on expectation the learning rule error is uncorrelated to the task gradient $\mathbb{E}[(\hat{\epsilon}_t^{lr})^T\nabla L_t] = 0$ (see S.I. 5.1 for full derivation). Equation (15), indicates that the change in task loss is facilitated by a steep slope (i.e. large $||\nabla L_t||$), and a small online learning error (large γ_t^o).

For the second term, we can show that

$$\mathbb{E}\left[\frac{1}{2}\delta\mathbf{w}_t{}^T\mathbf{H}_t\delta\mathbf{w}_t\right] = \frac{1}{2}\gamma_t^2\left(\nabla\hat{L}_t^o{}^T\mathbf{H}_t\nabla\hat{L}_t^o\right) + \frac{1}{2}(\eta_t^{lr})^2\mathbb{E}\left[(\hat{\epsilon}_t^{lr})^T\mathbf{H}_t\hat{\epsilon}_t^{lr}\right] \tag{16}$$

These terms capture the degree of curvature of the task loss landscape in the direction of $\nabla\hat{L}_t^o$ and $\hat{\epsilon}_t^{lr}$ (first and second term respectively) at the point in weight space $\mathbf{w}_t$. This quantity depends on the eigenvalues of the hessian $\mathbf{H}_t$ and how strongly $\nabla\hat{L}_t^o$ and $\hat{\epsilon}_t^{lr}$ project onto the different eigenvectors of the Hessian. We can make this relationship concrete and approximate the dependency of the hessian projection terms on the eigenvalues of the Hessian [54] (see S.I 5.2).

The learning rule error $\hat{\epsilon}_t^{lr}$ is uncorrelated with the task. Thus, on expectation it will project evenly onto the eigenvectors of the hessian (see box below). It follows that $(\hat{\epsilon}_t^{lr})^T\mathbf{H}_t\hat{\epsilon}_t^{lr}$ depends only on the average eigenvalue of the hessian (i.e. the average curvature)

$$\mathbb{E}[(\hat{\epsilon}_t^{lr})^T\mathbf{H}_t\hat{\epsilon}_t^{lr}] = \frac{Tr(\mathbf{H}_t)}{N} \tag{17}$$

,where N is the number of granule-cell-to-purkinje-cell weights and Tr denotes the trace of a matrix (see Fig 3 C) left).

Let ϵ be a random vector uncorrelated with the task loss. Let $\{\lambda_i, \hat{\mathbf{v}}_i\}, \quad i = 1, \ldots, N$ be the eigenvalue-eigenvector pairs of the hessian $\mathbf{H}_t$. By symmetry of $\mathbf{H}_t$, we can construct an orthonormal coordinate system from the eigenvectors of the hessian $\hat{\mathbf{v}}_i$. On expectation, it projects evenly onto all directions of the eigenvectors $\epsilon = \sum_{i=1}^N c\hat{\mathbf{v}}_i$. It follows that the projection onto the hessian

$$\hat{\epsilon}^T\mathbf{H}_t\hat{\epsilon} = \frac{\sum_{i=1}^N c\hat{\mathbf{v}}_i \sum_{i=1}^N \lambda_i\hat{\mathbf{v}}_i \sum_{i=1}^N c\hat{\mathbf{v}}_i}{\sum_{i=1}^N c\hat{\mathbf{v}}_i^T \sum_{i=1}^N c\hat{\mathbf{v}}_i} \tag{18}$$

$$= \frac{\sum c^2\lambda_i}{Nc^2} \tag{19}$$

$$= \frac{\sum \lambda_i}{N} \tag{20}$$

$$= \frac{Tr(\mathbf{H}_t)}{N} \tag{21}$$

where $Tr()$ denotes the trace of a matrix. In the last line we used an algebraic property of the relationship of the eigenvalues of a matrix and its trace. Hence, a random vector, uncorrelated with the task loss, projects evenly onto all the eigenvectors of the hessian. Its projection is onto the average curvature $\frac{Tr(\mathbf{H}_t)}{N}$.

Before looking at the projection of the online loss gradient ∇L_t^o, we consider the projection of the task loss gradient ∇L_t. Unlike the learning rule error, the task loss gradient ∇L_t won't project evenly onto the eigenvectors of the hessian $\mathbf{H}_t$. Consider the loss landscape near a local minimum $\mathbf{w}^*$ The loss landscape can be approximated by a quadratic function in the weights $\mathbf{w}$ and the hessian $\mathbf{H}_t$ is constant and positive definite. In this case, $\nabla L_t = \mathbf{H}_t(\mathbf{w}_t - \mathbf{w}^*)$. The vector $\mathbf{w}_t - \mathbf{w}^*$ can be decomposed in the directions of the eigenvectors of the Hessian $\mathbf{w}_t - \mathbf{w}^* = \sum_{i=1}^N c_i\hat{\mathbf{v}}_i$. Where $\{\lambda_i, \hat{\mathbf{v}}_i\}, \quad i = 1, \ldots, N$ are the eigenvalue-eigenvector pairs of the hessia ordered from smallest eigenvalue to the largest. Hence the projection of the gradient into the hessian can be written as follows

$$\nabla\hat{L}_t{}^T\mathbf{H}_t\nabla\hat{L}_t = \frac{(\mathbf{w}_t - \mathbf{w}^*)^T\mathbf{H}_t{}^T\mathbf{H}_t\mathbf{H}_t(\mathbf{w}_t - \mathbf{w}^*)}{(\mathbf{w}_t - \mathbf{w}^*)^T\mathbf{H}_t{}^T\mathbf{H}_t(\mathbf{w}_t - \mathbf{w}^*)} \tag{22}$$

$$= \frac{\sum_{i=1}^N c_i^2\lambda_i^3}{\sum_{i=1}^N c_i^2\lambda_i^2} \tag{23}$$

The hessian projection $\nabla\hat{L}_t{}^T\mathbf{H}_t\nabla\hat{L}_t$ can range from the smallest eigenvalue λ_1 (if $\mathbf{w}_t - \mathbf{w}$ is in the direction of the $\hat{\mathbf{v}}_1$) and λ_N (if it is is in the direction of $\hat{\mathbf{v}}_N$). If the weight state $\mathbf{w}_t$ is chosen at random in weight space, $\mathbf{w}_t - \mathbf{w}^*$ projects evenly onto the eigenvectors of the hessian. In that case, all the projection coefficients are equal

$c_i = c_j = c \quad \forall i, j = 1, \ldots, N$ and we get

$$\nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{\sum_{i=1}^N c^2 \lambda_i^3}{\sum_{i=1}^N c^2 \lambda_i^2} \tag{24}$$

$$= \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{25}$$

Note that in general, $\mathbf{w}_t$ is not chosen at random, it is determined by the weight trajectory from a particular learning rule. For example, if the learning rule is steepest gradient descent, the weights follow a trajectory that descends the steepest descent direction first. In that case, initially $\mathbf{w}_t - \mathbf{w}$ is expected to project evenly onto all the eigenvectors, throughout learning $\mathbf{w}_t - \mathbf{w}$ skews onto the directions with smaller curvature.

This gives us an explicit expression for the expected projection of the gradient ∇L_t. What about the online gradient $\nabla \hat{L}_t^o$? ∇L_t^o is an approximation of the task gradient ∇L_t based on the available window of time as in eq (7).

In one extreme case, the online gradient is an exact approximation of the task gradient $\nabla L_t^o = \nabla L_t$. In that case, the hessian projections are equal. The less correlated the online gradient and the task gradient are, the smaller we expect this projection to be. Indeed, the only way the projection would increase is if the online gradient projected more strongly onto the directions of large curvature. Yet that would imply knowledge of the Hessian of the task loss which is unlikely. Hence, we expect

$$\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o \leq \nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{26}$$

We can also bound the online gradient projection from the other side. Indeed, the other extreme case occurs when the online gradient is completely uncorrelated with the hessian. In that case, it projects evenly onto all the eigenvectors and $\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o = \frac{Tr(\mathbf{H}_t)}{N}$. Combining these two results we obtain the following bound

$$\frac{Tr(\mathbf{H}_t)}{N} \leq \nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o \leq \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{27}$$

We have shown this bound is reasonable near a local minimum, where the task loss landscape can be approximated by a quadratic function. In general, the task loss will be non-convex. Hence, it might have multiple local minima, maxima and saddle points. Let $\mathbf{w}^*$ be a local minimum in the neighborhood of $\mathbf{w}$. We can write the gradient of the task loss at $\mathbf{w}_t$ as follows

$$\nabla L_t = \mathbf{M}_t(\mathbf{w}_t - \mathbf{w}^*) \tag{28}$$

where

$$\mathbf{M}_t = \int_0^1 \nabla^2 L[\gamma(s)] ds \tag{29}$$

and $\gamma(s) = s\mathbf{w}^* + (1-s)\mathbf{w}_t, \quad s \in [0,1]$ is the parametrization of the straight line connecting $\mathbf{w}_t$ and $\mathbf{w}^*$. The vector ∇L_t can be decomposed in the directions of the eigenvectors of the hessian $\mathbf{M}_t(\mathbf{w}_t - \mathbf{w}^*) = \sum_{i=1}^N d_i \hat{\mathbf{v}}_i$. Hence the hessian projection of the gradient is

$$\nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{\sum_{i=1}^N d_i^2 \lambda_i}{\sum_{i=1}^N d_i^2} \tag{30}$$

We consider two cases. First, close to steady state i.e. when $||\mathbf{w}_t - \mathbf{w}^*||_2$ is small, then $\mathbf{M}_t \approx \mathbf{H}_t$ and $d_i = c_i \lambda_i$ and the argument from the quadratic task loss applies. The bounds in (27) still hold.

Second, as $||\mathbf{w}_t - \mathbf{w}^*||$ grows, i.e. further away from steady state, $\mathbf{M}_t$ is less similar than $\mathbf{H}_t$. If $\mathbf{M}_t$ and $\mathbf{H}_t$ were completely independent, then the projection of the gradient onto the different eigenvectors of the hessian is approximatively even and

$$\nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{\sum_{i=1}^N d_i^2 \lambda_i}{\sum_{i=1}^N d_i^2} = \frac{Tr(\mathbf{H}_t)}{N} \tag{31}$$

and the bounds in (27) are still valid.

Hence, for both linear and non-linear task loss functions and both at steady state and far from steady state the bounds in eq (27) are a good assumption. Where exactly $\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o$ falls in the bounds depends on multiple factors: the weight state $\mathbf{w}_t$ and how close it is to a local minima, and on γ_t^o —how close the online gradient direction $\nabla \hat{L}_t^o$ is to the task gradient direction $\nabla \hat{L}_t$ (see 5.4). $\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o$ increases with γ_t^o (i.e. as the online error η_t^o decreases).

We can say that $(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}$ depends on the *average curvature* (see fig 3 D)). As opposed to $\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o$ which depends on a *biased measure of curvature* of the task loss landscape. The average curvature determines how the learning rule error $\hat{\epsilon}_t^{lr}$ slows down learning (i.e. decreases the change in task loss), and the biased measure of curvature determines how the online gradient $\nabla \hat{L}_t^o$ does. We next examine how the network architecture, in particular, the input expansion modifies the slope, average curvature and biased measure curvature of the task loss landscape.

2.5 Effect of granule cell layer expansion on learning parameters

We have presented a model for online learning in the cerebellar cortex in the context of motor control. Online learning with a biologically plausible learning rule introduces two different types of error. We have shown how the geometry of the task loss landscape affects learning. In this section we show how the expansion from the mossy fibres to the granule cells affects the geometry of the loss landscape, in particular, the slope, the average curvature and the biased curvature.

Raman et al. showed that certain network expansions can increase learning performance in a theoretical setting [55]. To evaluate the effect of granule cell layer size, we look at a network expansion that adds granule cells. Let the original network have N granule cells and be in a network state determined by the output weights $\mathbf{w}$.

The network expansion maintains the number of mossy fibre inputs I and increases the number of granule cells from N to $\tilde{N}$. This increases the input expansion ratio from N/I to $\tilde{N}/I$. Let $q \equiv \tilde{N}/N$ be the expansion ratio, defined as the ratio between the input expansion ratios of the two networks. Each new granule cell forms connections with $K = 4$ randomly chosen input mossy fibres. Hence the input weight matrix of the expanded network $\tilde{Z} \in \mathbb{R}^{I \times \tilde{N}}$ is composed of the initial weights matrix Z and added columns each with K non-zero elements.

Each added granule cell forms a connection with the output purkinje cell. We assume the new connections initially have weight zero. This guarantees that the expansion preserves the input-output properties of the cerebellar-like network before learning. Indeed the task loss of the original and expanded network are equal $L[\mathbf{w}] = \tilde{L}[\tilde{\mathbf{w}}]$ but the granule cell layer activity $\mathbf{h}$ changes $\tilde{\mathbf{h}}(t) = \sigma(\tilde{Z}^T \mathbf{m}(t))$.

Consider the system at some time t, how are the online and task gradients modified with the expansion? The gradients depend both on the dynamics of the plant and the granule cell layer activity $\mathbf{h}$ from time 0 to t (see 5.3). The expansion of the granule cell layer doesn't modify the plant dynamics as the musculoskeletal system is unchanged. Hence, the change in the gradients from the expansion are determined by the change in $\mathbf{h}$.

Consider the expected granule cell activity $\mathbb{E}_t[\mathbf{h}(t)]$ over a uniform distribution of the time points in the interval $[0, t]$. As the expansion doesn't modify the mossy fibre activity, the expected norm of the granule cell activity of the expanded network scales linearly with the number of granule cells $\mathbb{E}_t[||\tilde{\mathbf{h}}(t)||^2] \propto \frac{\tilde{N}}{N} \mathbb{E}_t[||\mathbf{h}(t)||^2]$. Thus, we expect the norm squared of the gradients to grow linearly with the number of granule cells. We can write the scaling of the gradient norms as follows

$$||\nabla \tilde{L}_t|| = \alpha\,(q)\,||\nabla L_t||, \quad \alpha(q) = c_\alpha \sqrt{q} \tag{32}$$

$$||\nabla \tilde{L}_t^o|| = \beta\,(q)\,||\nabla L_t^o||, \quad \beta(q) = c_\beta \sqrt{q} \tag{33}$$

where $\nabla \tilde{L}_t = \nabla_{\tilde{\mathbf{w}}} \tilde{F}[\tilde{\mathbf{w}}]$ and $\nabla \tilde{L}_t^o = \nabla_{\tilde{\mathbf{w}}} \tilde{L}_{\Delta t_e}[t; \tilde{\mathbf{w}}]$ are the gradients in the expanded network and α and β are scalar functions that capture the scaling of the gradients as a function of the expansion ratio $q \equiv \tilde{N}/N$. The gradient norms increase with $\sqrt{q}$. Hence the slope of the task loss increases with the input expansion which facilitates learning.

Second, we consider the scaling of the hessian projection terms with the network expansion. In the previous section, we stated that the projection of the online gradient onto the Hessian can be bound by $Tr(\mathbf{H}_t)/N$ and $Tr(\mathbf{H}_t{}^3)/Tr(\mathbf{H}_t{}^2)$ as in equation (27) and the learning rule error projection only depends on $Tr(\mathbf{H}_t)/N$ (see (17)). Thus, it suffices to evaluate how these three terms are modified with the network expansion.

Like the gradient, the hessian depends on plant dynamics and the granule cell representation (see 5.3). Without knowing the plant dynamics we can't calculate directly the projections in the initial and expanded network or the hessian but we can consider how the quantities change relatively with the expansion (see 5.4). We find that the average eigenvalue stays constant $Tr(\tilde{\mathbf{H}}_t)/\tilde{N} = Tr(\mathbf{H}_t)/N$ but the spread of the eigenvalues grows at most linearly $\frac{Tr(\tilde{\mathbf{H}}_t{}^3)}{Tr(\tilde{\mathbf{H}}_t{}^2)} \leq \frac{\tilde{N}}{N} \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)}$.

Hence, the projection of the online gradient onto the hessian scales at most linearly with the network expansion.

$$\nabla \hat{\tilde{L}}_t^{oT} \tilde{\mathbf{H}}_t \nabla \hat{\tilde{L}}_t^o = \zeta\,(q)\,\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o \tag{34}$$

$$\zeta(q) = c_\zeta q^\rho, \quad 0 \leq \rho \leq 1 \tag{35}$$

As mentioned in the previous section, γ_t^o determines how close the projection is to the upper or lower bound. Hence, we expect ρ to increase with γ_t^o. The more online error there is, the less accurate the online gradient approximation and the smaller ρ is (see 5.4).

The hessian projection of the learning rule error $(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}$ is equal to the average eigenvalue of the hessian (see (17)) which remains constant with the expansion

$$\mathbb{E}[(\hat{\tilde{\epsilon}}_t^{lr})^T \tilde{\mathbf{H}}_t \hat{\tilde{\epsilon}}_t^{lr}] = \mathbb{E}[(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}] \tag{36}$$

We find that adding granule cells increases the slope of the task loss and the biased curvature but maintains the average curvature constant. What are the implications of the scaling of these terms on learning performance?

Quantity	Scaling function	Dependence on q
$\|\|\nabla \tilde{L}_t\|\|$	$\alpha(q)$	$\sqrt{q}$
$\|\|\nabla \tilde{L}_t^o\|\|$	$\beta(q)$	$\sqrt{q}$
$\nabla \hat{\tilde{L}}_t^{o\,T} \tilde{\mathbf{H}}_t \nabla \hat{\tilde{L}}_t^o$	$\zeta(q)$	$q^\rho, \quad 0 \leq \rho \leq 1$
$\mathbb{E}[(\hat{\tilde{\epsilon}}_t^{lr})^T \tilde{\mathbf{H}}_t \hat{\tilde{\epsilon}}_t^{lr}]$	1	q^0

Table 1: Summary of scaling of learning parameters with respect to network expansion ratio $q = \tilde{N}/N$.

2.6 Quantifying learning performance

We first examine how to quantify learning performance in the context of online motor learning. We can abstract learning to reducing some task loss $L[\mathbf{w}]$ by changing the parameters $\mathbf{w}$ of the system. If the system is learning, i.e. the weight update is beneficial, then ΔL_t is negative. At initial stages of learning, the synaptic weights are not optimised for the task hence the task loss is relatively large. The larger the magnitude of the change in task loss $|\Delta L_t|$, the larger the decrease in task loss. As learning goes on, the task loss decreases and the weights approach a local minimum in the loss landscape. Closer to a local minimum, the change in task loss becomes smaller. At steady state, by definition, the task loss is constant on expectation and the change in task loss fluctuates around zero.

Both learning speed and steady state performance are important in the context of motor learning. Indeed, if the environment or the plant change the internal model in the cerebellum is not well adjusted leading to a large task loss. The cerebellum must adapt to this change fast which requires large *learning speed*. At the same time, to achieve accurate movement performance, it must reach a good *steady state performance*.

We can define the learning speed and steady state loss with respect to the change in task loss ΔL_t. We define the learning speed for a change in weights $\mathbf{w}$ over a time window δt as the change in task loss per time:

$$\nu_t = -\frac{\Delta L_t}{\delta t} \tag{37}$$

Using the results from the previous section we can write the expected learning speed

$$\mathbb{E}[\nu_t] = \frac{1}{\delta t}\left[\gamma_t \gamma_t^o ||\nabla L_t|| - \frac{1}{2}\gamma_t^2 \nabla \hat{L}_t^{o\,T} \mathbf{H}_t \nabla \hat{L}_t^o - \frac{1}{2}\left(\frac{\gamma_t}{\sigma_t}\right)^2 \mathbb{E}\left[(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}\right]\right] \tag{38}$$

Note that the learning speed is only relevant at the initial stages of learning (when the task loss is relatively large). The learning speed is positive as long as the task loss is decreasing. At steady state, we expect the change in task loss ΔL_t to be small and to fluctuate around zero. We define the steady state loss ξ as the task loss when $|\Delta L_t| < \delta$, for some small δ.

Both of learning speed and steady state loss are defined with respect to ΔL_t. Given the system at time t with state $\mathbf{w}_t$ and task loss L_t, the learning speed from $\delta \mathbf{w}_t$ is maximised when $|\Delta L_t|$ is maximised. For the steady state loss, the relationship is not as straightforward (see box below). The local task difficulty is defined as

$$G_t \equiv \frac{1}{2}\frac{\gamma_t}{||\nabla L_t||}\left[\nabla \hat{L}_t^{o\,T} \mathbf{H}_t \nabla \hat{L}_t^o + \frac{\mathbb{E}\left[(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}\right]}{\sigma_t^2}\right] \tag{39}$$

The local task difficulty G_t determines the steady state loss ξ. The smaller the local task difficulty, the smaller the steady state loss (see [55]). Consider a system at steady state with loss $\xi = L_{SS}$. From the box bellow, at steady state the inequality (42) is broken. If the local task difficulty were reduced while maintaining the left hand side γ_t^o (for example with a network expansion) then the system would keep learning reducing the steady state loss $\tilde{\xi} \leq \xi$.

The subscript t in (39) and (38) represents the time during the trajectory at which we evaluate each quantity. Although we can evaluate ν_t and G_t for any t, these quantities are relevant for different regimes of learning. ν_t is relevant at initial stages of learning when L is large, and G_t will be a proxy of steady state loss if it is evaluated at steady state, when L is small and the weight change is as well. In these different regimes, some of the quantities, like γ_t^o, $||\nabla L_t||$ and $\nabla \hat{L}_t^{o^T} \mathbf{H}_t \nabla \hat{L}_t^o$, have different values.

Steady state loss analysis: Consider the expected change in task loss $\mathbb{E}[\Delta L_t]$, at some time t given a change in weights $\delta \mathbf{w}_t$. We can rewrite the expression in eq (14) as follows

$$\mathbb{E}[\Delta L_t] = \gamma_t ||\nabla L_t|| [-\gamma_t^o + \frac{1}{2}\gamma_t \frac{1}{||\nabla L_t||} \nabla \hat{L}_t^{o^T} \mathbf{H}_t \nabla \hat{L}_t^o \tag{40}$$

$$+ \frac{1}{2} \frac{(\eta_t^{lr})^2}{\gamma_t} \frac{\mathbb{E}\left[(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}\right]}{||\nabla L_t||} \Bigg] \tag{41}$$

Learning requires a negative change in task loss, and it stops when

$$\gamma_t^o \leq \frac{1}{2}\gamma_t \frac{1}{||\nabla L_t||} \nabla \hat{L}_t^{o^T} \mathbf{H}_t \nabla \hat{L}_t^o + \frac{1}{2} \frac{(\eta_t^{lr})^2}{\gamma_t} \frac{\mathbb{E}\left[(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}\right]}{||\nabla L_t||} \equiv G_t \tag{42}$$

We refer to the right hand side of this equation as the local task difficulty based on the definition by Raman et al. [55]. This inequality is bound to be satisfied as when the task loss approaches a local minimum $||\nabla L_t||$ goes to zero, $\mathbf{H}_t$ becomes positive definite hence the hessian projections $\nabla \hat{L}_t^{o^T} \mathbf{H}_t \nabla \hat{L}_t^o$ and $(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}$ are positive.

In the ideal learning scenario, the task loss is reduced to zero in one iteration and stays zero (one shot learning). This would correspond to zero steady state loss and maximum learning speed. However, in most learning scenarios this level of performance is not achievable as it would require exact knowledge of the geometry of the loss surface by the learning rule. In practice, there is a trade-off between learning speed and steady state loss.

2.7 Trade-off between learning speed and steady state loss

Given a direction of weight plasticity $\delta \hat{\mathbf{w}}_t$, we can find the optimal plasticity magnitude $||\delta \mathbf{w}_t||^*$ that minimises the change in task loss ΔL_t given in eq (14) (see 5.5 and [54])

$$||\delta \mathbf{w}_t||^* = \frac{-\delta \hat{\mathbf{w}}_t^{\,T} \nabla \hat{L}_t}{\delta \hat{\mathbf{w}}_t^{\,T} \mathbf{H}_t \delta \hat{\mathbf{w}}_t} ||\nabla L_t|| \tag{43}$$

(as long as $\delta \hat{\mathbf{w}}_t^{\,T} \nabla \hat{L}_t \leq 0$, and $\delta \hat{\mathbf{w}}_t^{\,T} \mathbf{H}_t \delta \hat{\mathbf{w}}_t > 0$). The optimal plasticity magnitude depends linearly on the magnitude of the gradient (i.e. the slope of the task loss surface) and is decreased by the projection of $\delta \hat{\mathbf{w}}_t$ on the hessian of the task loss. In general, far from a local minimum (i.e when the gradient is large with respect to the curvature) the optimal plasticity magnitude is large. During learning, the gradient norm decreases with respect to the curvature hence the optimal plasticity magnitude decreases. Close to a local minimum, a large change in weights can lead to overshooting preventing a decrease in task loss (see Fig 4 A)). Machine learning algorithms performing gradient descent use this fact. For example, in stochastic gradient descent γ should decrease linearly with the number of iterations of training [9, 37].

In our learning model given by eq (9), the synaptic plasticity is composed of the online gradient term $\gamma_t \nabla \hat{L}_t^o$ and learning rule error term $\eta_t^{lr} \hat{\epsilon}_t^{lr}$. γ_t converts the feedback information on the loss ∇L_t^o into the weight change $\delta \mathbf{w}_t$. This quantity might vary from synapse to synapse but we can think of γ_t in our formula as a population average. Given some σ, there is an optimal γ_t^* (see box below).

If at each step of learning, the γ_t^* is applied, then both learning speed and steady state loss can be optimised. However, in practice, it is computationally expensive to calculate γ_t^* as it requires calculating second order loss terms at each step. Biologically, it is unlikely that the system can compute γ_t^*, as it requires knowledge of the gradient and hessian of the task loss as well as learning rule error, all of which are unknown. It is more plausible that γ is constant throughout learning. In that case, the question arises, how large should γ be for optimal learning?

On one extreme, if γ is too large it can lead to learning instability (i.e. task loss doesn't decrease or reach a steady state value). If γ is small enough to assure learning stability, then we find a trade-off between learning speed and steady

state loss (see Fig 4 A)). A large γ, means the system is very sensitive to new information and can learn fast but cannot achieve good steady state performance. On the other hand, a small γ leads to a slow learning speed but better steady state performance. Indeed, for small γ, the learning speed increases with γ and so does the local task difficulty G_t. Increasing γ can increase the learning speed but at the expense of decreasing steady state performance (see box).

Optimal γ: Given a direction of plasticity as in equation 9, with fixed $\sigma = \frac{\gamma_t}{\eta_t^{lr}}$ we can find γ_t^* is such that $||\delta \mathbf{w}_t||^* = -\gamma_t^* \nabla L_t^o + \frac{\gamma^*}{\sigma}\hat{\epsilon}_t^{lr}$. Using this with eq (43) we get

$$\gamma_t^* = \frac{\gamma_t^o ||\nabla L_t||}{\nabla \hat{L}_t^{o^T} \mathbf{H}_t \nabla \hat{L}_t^o + \frac{1}{\sigma^2}(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}} \tag{44}$$

If σ is large (i.e. small learning rule error), then the second term in the square root is small and we get $\gamma_t^* \approx \frac{\gamma_t^o ||\nabla L_t||}{\nabla \hat{L}_t^{o^T} \mathbf{H}_t \nabla \hat{L}_t}$. The learning rule error effectively decreases γ_t^*.
At initial stages of learning, $||\nabla L_t||$ is relatively strong compared to the hessian projections, and decreases with learning. Hence γ^* is large initially and decreases as the task loss decreases.
Close to a local minimum, as the task gradient converges to zero, the γ^* also converges to zero. Indeed, if the weights were at the exact local minimum the optimum change in weights would be zero. In practice, however, it is unexpected for a learning rule with errors to converge perfectly to the local minimum. Hence, γ^* in practice will be non-zero.
In general, the γ that optimises learning speed γ_ν^* is larger than the one that optimises steady state error γ_ξ^*.
For a fixed γ over learning, there are two regimes

- $\gamma_\xi^* \leq \gamma \leq \gamma_\nu^*$: there is a trade-off between learning speed and steady state loss. Increasing γ makes steady state loss worse but learning speed better.
- $\gamma_\xi^* \leq \gamma_\nu^* \leq \gamma$: both learning speed and steady stater error get worse, learning can reach instability

As in the second case there might not be any learning because of instabilities, the first case is the relevant one for this analysis.

2.8 Granule cell expansion navigates the trade-off between learning speed and steady state loss

In the previous section, we found that having a constant γ throughout learning introduces a trade-off between learning speed and steady state loss. Furthermore, we found in 2.5 that increasing the granule cell expansion modifies the loss landscape. In this section, we link these two results together and consider how the granule cell layer expansion affects the trade-off between learning speed to steady state loss.

First, we look at how the granule cell layer expansion modifies the learning speed in (37). As the network expansion described above conserves the task loss L_t and the time window δt, the scaling of the learning speed only depends on the scaling of ΔL_t. We can write the learning speed in the expanded network using the results in table 1.

$$\mathbb{E}[\tilde{\nu_t}] = \frac{1}{\delta t}\left[\tilde{\gamma}\gamma_t^o \alpha(q)||\nabla L_t|| - \frac{1}{2}\tilde{\gamma}^2 \zeta(q) \nabla \hat{L}_t^{o^T} \mathbf{H}_t \nabla \hat{L}_t^o - \frac{1}{2}\left(\frac{\tilde{\gamma}}{\sigma}\right)^2 \mathbb{E}\left[(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}\right]\right] \tag{45}$$

Where we have used the fact that $\tilde{\gamma}_t^o = \gamma_t^o$ and $\tilde{\sigma} = \sigma$. These coefficients are defined in the expression for $\delta \mathbf{w}_t$ in eq (11) and don't depend on the network architecture, only on the learning rule and the limitations form online learning. Hence, we expect these coefficients to remain independent of the network expansion. For instance, γ_t^o depends on the online error time window $\Delta t_e - \Delta t_r$ and the frequency components of the reference signal which don't change as the granule cell layer expansion is modified.

The learning speed in the expanded network depends on $\tilde{\gamma}$ in the expanded network. γ represents the relationship between the error information carried through climbing fibres and the resulting synaptic plasticity. We can't rule out the possibility that neural circuits with different expansion ratios from mossy fibres to granule cells have different γ. Indeed, this quantity could be regulated at the circuit level. We hence assume that γ might be different for different input expansions.

We can show that given the initial network with γ, we can find a $\tilde{\gamma}$ in the expanded network that improves learning speed $\tilde{\nu}_t[\tilde{\gamma}] \geq \nu_t[\gamma]$ (see box below). The γ in the expanded network must be smaller than in the initial network but must not be too small.

Learning speed scaling: We can show that there is mapping from $\gamma \rightarrow \tilde{\gamma}$ that guarantees an improvement in the learning speed. The first term in $\mathbb{E}[\tilde{\nu}_t]$ eq (45) increases if and only if

$$\tilde{\gamma} > \frac{\gamma}{\alpha(q)} \tag{46}$$

The second term decreases if and only if

$$0 < \tilde{\gamma} < \frac{\gamma}{\sqrt{\zeta(q)}} \tag{47}$$

And the third term decreases if and only if

$$0 < \tilde{\gamma} < \gamma \tag{48}$$

From the table 1, we get that the scaling of $\alpha(q) \sim q^{1/2}$ and $\sqrt{\zeta(q)} \sim q^{\rho/2}$. Yet, $0 \leq \rho \leq 1$ and $q \geq 1$, hence $q^{-1/2} \leq^{-\rho/2} \leq 0$.
Thus,

$$\text{If } \frac{\gamma}{\alpha(q)} \leq \tilde{\gamma} \leq \frac{\gamma}{\sqrt{\zeta(q)}} \quad \text{then } \mathbb{E}[\tilde{\nu}_t] \geq \mathbb{E}[\nu_t] \tag{49}$$

with the equality happening if $\rho = 1$.

Second, we can express the local task difficulty as defined in (39) in the expanded network

$$\mathbb{E}[\tilde{G}_t] = \frac{1}{2}\tilde{\gamma}\frac{1}{||\nabla L_t||}\nabla\hat{L}_t^{o^T}\mathbf{H}_t\nabla\hat{L}_t^o\frac{\zeta(q)}{\alpha(q)} + \frac{1}{2}\frac{\tilde{\gamma}}{\sigma^2}\frac{\mathbb{E}\left[(\hat{\epsilon}_t^{lr})^T\mathbf{H}_t\hat{\epsilon}_t^{lr}\right]}{||\nabla L_t||}\frac{1}{\alpha(q)} \tag{50}$$

From section 2.5, we get that $\frac{\zeta(q)}{\alpha(q)} \sim q^{\rho-1/2}$ so the first term increases with the expansion due to the increase in the overall curvature of the loss landscape. The second term, on the other hand, decreases with the expansion as the norm of the task gradient increases with the expansion and the average curvature is constant. Hence, the granule cell layer expansion decreases the effect of the learning rule error on learning performance.

We can find a $\tilde{\gamma}$ in the expanded network such that the first term is also reduced with the network expansion (see box).

Steady state loss scaling: The first term in the local task difficulty (50) decreases with the expansion if and only if

$$\tilde{\gamma} < \frac{\alpha(q)}{\zeta(q)}\gamma \sim q^{1/2-\rho} \tag{51}$$

and the second term decreases with the expansion if and only if

$$\tilde{\gamma} < \alpha(q)\gamma \sim q^{1/2} \tag{52}$$

Putting these two equations together we get

$$\text{If } \tilde{\gamma} < \frac{\alpha(q)}{\zeta(q)}\gamma, \quad \text{then } \mathbb{E}[\tilde{\xi}] \leq \mathbb{E}[\xi] \tag{53}$$

We can find γ in the expanded network such that the steady state loss is reduced by the network expansion.

We have shown that adding granule cells can improve both learning speed and steady state loss by modifying the loss geometry in a beneficial way for online learning. No matter what the input expansion —the ratio between number of mossy fibres and granule cells— is in a cerebellar cortex module, there is a trade-off between learning speed and steady state loss regulated by γ. However, increasing the input expansion by adding granule cells shifts the trade-off such that the larger network can achieve better learning speed and steady state loss simultaneously.

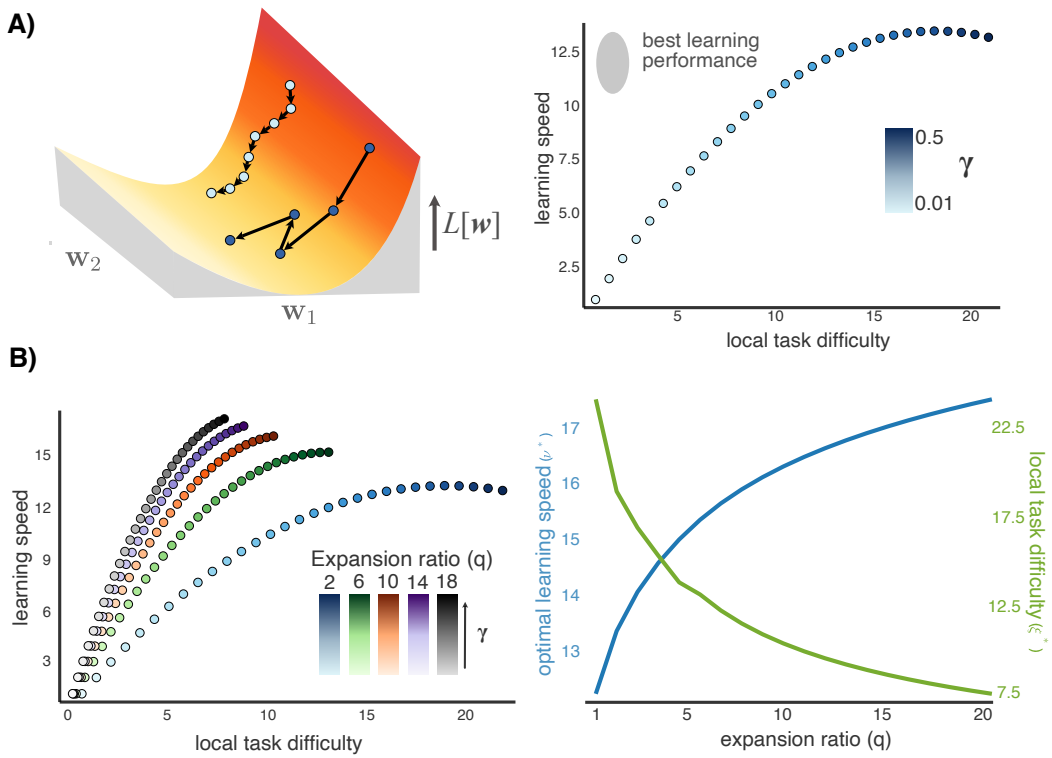

Figure 4: Granule cell expansion navigates the trade-off between learning speed and steady state loss. **A)** The strength of γ represents the sensitivity of the learning rule to new information. This quantity modulates a trade-off between learning speed and steady state loss. On the left, visualisation of a weights trajectory during learning for a small γ (light blue) and large γ (dark blue). A small γ descends the loss landscape slowly (small learning speed) but can get closer to a local minimum (small steady state loss). A large γ leads to large steps that descend the loss landscape fast but oscillate around the local minimum. On the right, steady state loss vs learning speed for different values of γ. **B)** Adding granule cells mitigates the trade-off between learning speed and steady state loss. On the left, steady state loss vs learning speed for different network expansion values $q = \tilde{N}/N$. As the number of granule cells increases the system can achieve better learning speed for the same steady state performance. On the right, the steady state loss (in orange) for the maximal learning speed (in blue) decreases with q, hence better learning speed and steady state loss can be achieved simultaneously by increasing the input expansion.

Combining the $\gamma \to \tilde{\gamma}$ mappings to improve learning speed and local task difficulty, we get a formula for the mapping that improves both simultaneously

$$\text{If } q^{-1/2} \sim \frac{\gamma}{\alpha(q)} \leq \tilde{\gamma} \leq \gamma \frac{1}{\sqrt{\zeta(q)}} \sim q^{-\rho/2} \quad \text{then } \tilde{\nu} \geq \nu, \quad \tilde{\xi} < \xi \tag{54}$$

So increasing the input expansion and reducing γ can take advantage of the increase of the slope of the loss landscape and harness the effect from the increase of the biased curvature to improve learning performance. In theory, the learning performance just keeps increasing as the number of granule cells increases. However, in practice, at some point, the γ needed to improve learning performance will be too small to be plausible biologically. Indeed, such a small change in synaptic plasticity could be out-weighted by synaptic noise [55]. This will provide a practical bound on the minimal γ and hence on the maximum input expansion.

The bounds for the mapping are determined by the scaling functions of the gradient α and of the hessian projection of the online gradient ζ. We mention in section 2.5 and 5.4 that the scaling of the hessian projection depends on the online learning error. Larger online learning error in general leads to smaller ρ. This means that the worse the online learning error is, the less γ needs to decrease to improve learning performance. This will permit large expansions before the biological constraints on γ are reached. The input expansion in the cerebellar circuit is even more beneficial when the online gradient is less correlated with the task gradient (see 5.4).

Note that we showed that the granule cell expansion can increase learning performance but we haven't quantified by how much. This will again depend on the parameters of the problem. In particular, the less accurate a learning rule is (i.e. smaller σ), the more benefit there is in adding granule cells to decrease steady state loss.

We find that the granule cell expansion relaxes the trade-off between learning speed and steady state loss. The change in the geometry of the loss landscape from the expansion allows for better learning speed for the same steady state loss. Conversely, it can also lead to better steady state loss for the same learning speed (see Fig 4 B)).

3 Discussion

We have explored the relationship between the structure of excitatory feedforward networks in the cerebellar cortex and their ability to perform online motor learning. We formalised the constraints of learning while executing motor control, and examined how the expansion from mossy fibres inputs to granule cells affects learning performance. Our results show that the divergent circuit structure in the cerebellar cortex is necessary for fast and accurate motor learning. Increasing the input expansion navigates the trade-off between learning speed and accuracy while learning in real-time to control a musculo-skeletal system. Our work suggests that large input expansion can mitigate the limitations from learning fast and with limited information.

The study of the structure of cerebellar-cortex circuits in relationship to the cerebellar function has a long history. 50 years ago, Marr and Albus proposed a computational model of cerebellar cortical function in which the granule cell layer separates input activity patterns [40, 1]. Following work has broadened the concept of expansion recoding in cerebellar networks for pattern separation [38, 15, 16, 69, 8, 68, 60]. However, it is not clear that the cerebellum only performs pattern separation. Studies of granule cell activity suggest that they might not always act as sparse coding pattern discriminators [30]. Our work expands the study of cerebellar circuits beyond pattern separation to the study of a dynamic task such as motor learning. Furthermore, previous work on the role of expansion recoding analysed either learning speed of a downstream classifier [15] or the performance of a Hebbian classifier [38]. In our work, we consider simultaneously learning speed and steady state loss performance. We find that in a supervised learning with fixed γ, there is a trade-off between the two quantities. Larger sensitivity to error information (i.e. larger γ) leads to faster learning but limits accuracy (see Fig 4 A)).

The classical theories of expansion recoding in the cerebellum relied on the sparse coding in the granule cell layer. Indeed, they required only a few granule cells to be active at a time (per pattern) [40, 1, 69, 60, 8, 2]. In the last five years, sparse coding has been put into question. Experimental studies have found that the granule cell population activity is much denser than previously believed [3, 35, 23, 71, 22]. Furthermore, some work has showed that sparse coding has some theoretical limitations and trade-offs with the generalisation ability [64]. These findings rekindled theoretical work on the granule cell representation for pattern separation [15, 16, 38]. The work found that population sparsity might not be necessary for efficient pattern separation. In fact, decorrelation at the granule cell layer is the main determinant of pattern separation performance. Our findings didn't require sparse coding for motor learning. In fact, we didn't constrain the coding level at the granule cell layer, we just set a sparse connectivity from mossy fibres to granule cells. Our results provide a link between cerebellar architecture and function that is consistent with experimental findings of coding level at the granule cell layer.

The execution of fast coordinated movements is too difficult to be solved by simple feedback control due to the delays in feedback loops and small gains. Internal models capture the input-output relationships of processes, such as the dynamics of a musculoskeletal system, in neural circuits. These models are used to make predictions about the motor apparatus and help with the execution of fast, smooth and coordinated movements [73, 33, 6]. It is widely accepted that the cerebellum acquires internal models of the motor apparatus for motor control. However, the exact organization of internal models is still under debate [33, 50]. We chose a model in which the cerebellum learns and inverse model, the input being the aimed trajectory and the output a motor command. It is also believed that the cerebellum can learn a forward model, that maps an efference copy of the motor command to the estimate of the plant state [73, 50, 51, 28, 49, 43]. In fact, both forward models and inverse models can be viewed as inter-related. Acquisition of a motor act requires forward models but the acquisition process itself requires an inverse model. In the literature, we find theories of motor control that include both forward and inverse models in the cerebellum working together [73]. Although we presented our results for a system with an inverse model, we expect the results to still apply for a forward model. Indeed, acquiring a forward model in the context of motor control, still has the limitations from online learning. So although the signals are different and the loss function is as well, the decomposition of the change in weights 11 and the effect on the loss landscape from the expansion still applies.

We modelled the cerebellar-like network as three-layer feedforward network with a single purkinje cell output. This description is a simplification which omits interneurons and deep cerebellar nuclei. Golgi cells are excited by mossy fibres and parallel fibres and inhibit granule cells [19, 29]. Molecular layer interneurons provide inhibitory influence to

purkinje cells' dendrites which can have a strong effect on the cerebellar cortex computations [10, 39]. Inhibition in the input layer of the cerebellar cortex can decorrelate and sparsen granule cell activity [38, 8]. We also omitted deep cerebellar nuclei which receive inhibitory inputs from Purkinje cell and collaterals from mossy fibres. It would be of value to asses how are results apply in a more detailed model of the cerebellar circuits. However, our model underlines the primary features of cerebellar information processing: the processing of inputs thanks to the vast divergence from mossy fibres to granule cells.

We chose model with rate coding and a simple musculoskeletal system (linear plant). This is clearly a simplification of the biological properties, it was chosen to make the analytical results more tractable. There are many instances of motor control models that include spiking cerebellar cortex models and more complex musculoskeletal systems [70, 14, 13]. These models show that even with a spiking model and a complex plant, the motor control system we presented here can learn. Furthermore, we assumed that most of the learning in the cerebellar cortex originates in the plasticity of the parallel fibre to Purkinje cell synapses. There are other sites of plasticity in the cerebellum, however, those sights provide weaker and slower tuning of cerebellar outputs. Indeed, each Purkinje cell dendrites may form as many as 200,000 synapses providing a huge basis of synaptic plasticity [39].

We found that learning online in the cerebellum introduces constraints on the amount of information carried through the climbing fibre and hence on the learning in the cerebellar cortex. We assumed that the climbing fibre carries some signal correlated with the online task loss. In the first section we examined simulations with a local learning, LMS-like learning rule, but in the rest of the analysis we remained quite agnostic of the exact learning rule. Cerebellar learning can be quite complex. Ito proposed that parallel-fibre-to-Purkinje-cell synapses undergo long-term depression (LTD) when climbing fibres and parallel fibres are activated [27]. Long-term potentiation (LTP) was later discovered to lead synaptic plasticity in the absence of climbing fibre inputs [25]. Since then, multiple forms of plasticity have been found at almost every synapse in the cerebellum circuit [12, 61, 56]. It would be of great interest to apply our results to a model with a more biologically detailed learning rule.

Although classically the function of the cerebellum is considered to be purely motor-related, recent imaging studies have shown the cerebellum to be involved in non-motor related cognitive functions [48, 59, 18, 4, 36, 46, 26]. It is believed that the role of the cerebellum in these cognitive tasks is similar to its role in motor tasks. The cerebellum would act as a predictive machine which learns internal models of the environment, musuculo-skeletal system or cerebral neural circuits with supervised learning [20, 26]. Depending on the inputs and outputs received by a cerebellar module it would be used for different tasks such as motor control or cognitive processes [26, 45, 73]. In this work we examined the case of motor control, it would be interesting to study the effect of the network expansion on online learning in a non-motor system.

4 Materials and Methods

Full details of all simulations are provided in the *Supporting Information*. In this section we provide an overview. All the simulations were done on Julia.

4.1 Motor control system

We model the motor control system in fig 1 C).

The feedback controller is modelled as a PID controller with gains $K_p = K_d = K_i = 0.1$. The output is given by $u_{fb}(t) = K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}$, where $e(t) = r(t) - y(t)$ is the trajectory error.

The musculoskeletal system is modelled as a single input single output stable linear plant with the space state model

$$\begin{cases} \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu \\ y(t) = C\mathbf{x}(t) \end{cases} \tag{55}$$

where

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -2 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -2 & -2 \end{pmatrix} \tag{56}$$

$$B = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} \tag{57}$$

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \tag{58}$$

For simulations we chose the initial conditions to be all zeros.

The reference trajectory is the $r(t), t \in [0, T]$ that the system needs to learn to track. In any system there are signals that the plant won't be able to produce because of the limitations of its dynamics. Biologically, this makes sense as there are some movements that will be too fast or to big for the musculoskeletal system to produce. We can express this limitation by a cut-off frequency f_c. We assume that any signal that doesn't have any frequencies above f_c, can be actuated by the plant. We chose $f_c = 0.1Hz$. We want an unbiased reference signal that can be learned by the system.

We define the reference signal from a Fourier series as follows

$$r(t) = \frac{c_0}{2} + \sum_{k=1}^{N_{ref}/2} c_k \sin(2\pi f_k t + \phi_k) \tag{59}$$

where N_{ref} is the number of different sinusoidals used to generate the signal. We choose by default $N_{ref} = 200$. The harmonic frequencies are

$$f_k = \frac{2kf_c}{N_{ref}}, \quad \text{for } k = 1, 2, ..., \frac{N_{ref}}{2} \tag{60}$$

This satisfies that the highest harmonic frequency matches the cut-off frequency f_c. The phase shifts ϕ_k are drawn from a random uniform distribution over the interval $[0, 2\pi)$. The constant coefficients c_k are selected according to the desired power spectrum. We want to approximate a filtered Ornstein-Uhlenbeck (O-U) process. The power spectrum of an O-U process is a Lorentzian (*Mathematics for Neuroscientists*). Hence we draw the c_k coefficients from a Lorentzian

$$c_k = \frac{c\tau}{2} \frac{2\tau}{1 + (2\pi\tau k)^2}$$

where τ is the inverse of the width and c controls the magnitude of c_k.

We want $|r(t)| < 1$ to make sure that the plant can realize the reference trajectory. Yet the steady state variance of an O-U process with coefficients c and τ is $\frac{c\tau}{2}$. We chose $\tau = \frac{1}{N_{ref}}$ and $c = \frac{2\pi}{N_{ref}}$. Increasing τ, decreases the strength of higher frequencies in the signal. Increasing c, increases the overall magnitude (variance) of $r(t)$.

The inputs to the cerebellar-like network $\mathbf{m}(t)$ are defined as in eq (1), where $\Delta t_I = 0.2/f_c$ and $\delta t_I = \Delta t_I/(I-1)$ where I is the number of inputs to the cerebellar like network. Learning is relatively sensitive to the choice of Δt_I; if it is too small, there is no enough lookahead for the cerebellar like network to learn the inverse model. If δt_I is larger than one quarter of the period of the largest frequency sinusoidal, the inputs $\mathbf{m}$ don't reflect the dynamics of the reference signal hence learning night not happen. We define the length of the lookahead window Δt_I with respect to the cut-off frequency ω_c to make sure that the lookahead window is large enough to include some variation in the reference trajectory. Our results are robust to the choice of these parameters as long as the lookahead window includes a non-constant portion of the reference trajectory $[r(t), r(t + \Delta t_I)]$ but no more than quarter of the period of the largest frequency sinusoidal.

Cerebellar-like network For simulations, we initialise the network architecture to have $I = 10$ inputs, a single output and the number of granule cells N varies. Each granule cell passes inputs through a hyperbolic tangent nonlinearity $\tanh : \mathbb{R} \to \mathbb{R}$ of the form $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. The purkinje cells have a linear output.

Each granule cell receives $K = 4$ inputs from randomly chosen mossy fibres. The input weights are drawn from standard-normal distribution with mean 0 and standard deviation $1/\sqrt{K}$. Hence, the input weight matrix Z has N columns with each column having K non-zero elements. The input weights are constant throughout the trajectory. The granule cell layer to purkinje cell layer is fully connected. Each output weights is drawn from a standard normal distribution with mean zero and standard deviation $1/\sqrt{N}$. The output weights are plastic and are updated according to a learning rule.

Learning rule The goal of this paper is not to determine learning rules in the cerebellar cortex for motor control. However, we test different learning rules to to determine the validity of our model. The weights are updated throughout the trajectory every $\delta t_e = 1$, we set this interval to be uniform for convenience. We use an adapted version of least mean squares (LMS) [72]. The weight update at each update time is given by (5). The vector of learning rule error $\hat{\epsilon}^{lr}$, each element is drawn independently from a random normal distribution $\mathcal{N}(0, 1)$.

The classic LMS rule is given by $\delta \mathbf{w}_t = -\gamma \mathbf{h}(t)(r(t) - y(t))$. The main differences between our learning rule and classic LMS are

- In classic LMS the weight update is continuous, in our case, the weights are updated at discrete points in time during the trajectory.
- The climbing fibre activity encodes the trajectory error over the time window $[t - \Delta t_e - \Delta t_r, t - \Delta t_r]$, instead of the instantaneous error. Δt_r denotes the delay between the trajectory is executed and the climbing fibre signal arrives to the cerebellar network.
- We introduce a possible delay between the granule cell activity and the climbing fibre activity leading to synaptic plasticity. This delay is represented by Δt_h (see [67]).
- We include learning noise $\eta \hat{\epsilon}_t$ arising from imperfections in the synaptic plasticity mechanisms or in the information on the error $e(t)$.

Unless otherwise specified we set $\Delta t_e = 1.0$, $\Delta t_r = 0.5$ and $\Delta t_h = 0.1$.

4.2 Network expansion

The network expansion maintains the number of mossy fibre inputs I and increases the number of granule cells from N to $\tilde{N}$. This increases the input expansion ratio from N/I to $\tilde{N}/I$. Let $q \equiv \tilde{N}/N$ be the expansion ratio, defined as the ratio between the input expansion ratios of the two networks. Each new granule cell forms input connections with $K = 4$ randomly chosen input mossy fibres. Hence the input weight matrix of the expanded network $\tilde{Z} \in \mathbb{R}^{I \times \tilde{N}}$ is composed of the initial weights matrix Z and added columns each with K non-zero elements.

$$\tilde{Z} = \left(\quad Z \quad \middle| \begin{matrix} | & & | \\ \tilde{\mathbf{z}}_{N+1} & \dots & \tilde{\mathbf{z}}_{\tilde{N}} \\ | & & | \end{matrix} \right) \tag{61}$$

Each added granule cell forms a connection with the output purkinje cell. We assume the new connections initially have weight zero so the initially don't contribute to the input-output properties of the network. The output weights of the expanded network are

$$\tilde{\mathbf{w}} = \begin{pmatrix} -\mathbf{w}^T- & | & 0 \end{pmatrix}^T \tag{62}$$

This expansion preserves the input-output properties of the cerebellar-like network before learning. Indeed the task loss of the original and expanded network are equal $L[\mathbf{w}] = \tilde{L}[\tilde{\mathbf{w}}]$ but the granule cell layer activity $\mathbf{h}$ changes $\tilde{\mathbf{h}}(t) = \sigma(\tilde{Z}^T \mathbf{m}(t))$.

4.3 Details for simulations in motivating example

We simulate the system as described in 4.1 for a trajectory time $T = 1000$. At each time of weight update t, the task loss $L[\mathbf{w}_t]$ is computed. To do so, a system identical to the one being trained, except with fixed output weights $\mathbf{w}$ (no weight update) is defined. We compute the task loss as defined in eq 6 in this static system. Indeed, the task loss reflects the performance of the output weights $\mathbf{w}$ on the task, over the whole trajectory. The task gradient ∇L_t is also computed in the static system.

For Figure 2, we fit the task loss $L[\mathbf{w}_t]$ with an exponentially decaying function $f(t) = c_1 e^{-c_2 \cdot t} + c_3$. We take the **learning speed** $\nu = c_2$.

To compute the **steady state loss** we pre-train the network with gradient descent to a very low task loss (we set $T = 7000$ and $\gamma = 0.1$ for pre-training). The weights after pre-training are close to the optimal weights. We use these weights as initial weights and train with the LMS-like learning rule. During this training, the task loss initially increases -as the initial weights are very close to the local minimum- and then settles around a steady state value. We set the steady state loss to the mean of the task loss over the last 100 steps. The pre-training is important to recover the optimal steady state value without having to train the network with the LMS-like learning rule for a very long trajectory time. In the absence of pre-training the systems, especially trained with small γ might not reach steady state.

We compute the learning speed and steady state loss values for a range of number of granule cells $N = 10, \ldots, 90$ and a range of γ. We perform the network expansion as described in 4.2. For each network expansion explore a different range of γ indeed, larger networks in general require smaller γ for learning. For Figure 2, we have $\gamma \in \frac{1}{\sqrt{q}}[0.01, 0.02, \ldots, 0.1]$. We normalise the values of the learning speed and steady state loss by the values for the smallest q and γ to emphasize the relative changes.

For Figure 2, we performed 25 simulations with different reference trajectories and different weight initialisations for the cerebellar like network. We selected the simulations with similar initial task loss (loss before learning). In particular, we discard simulations with initial loss below the average initial loss minus $1/2$ standard deviation. This filters out initialisations with very low initial loss that might not converge when trained with the upper range of γ. After filtering out we are left with 17 simulations.

For Figures 2 C) and D),for each simulation, and for each q, we compute the optimal learning speed over all γ and the steady state loss value for the same value of γ. Each point represents the mean over all simulations and the error bars represent the standard error.

Figure 2 B) represents the trade-off between steady state loss and learning speed over γ for a single simulation. For one reference trajectory, we train for all N and γ starting at the same weight initilisation 20 different times. The learning rule error $\nu^{lr}\hat{\epsilon}_t^{lr}$ in the weight update (5) introduces a slight variability each training simulation. We plot the learning speed vs steady state loss for $q = 1, 7$, for all γ and for all training simulations. We plot the trend line for each q.

For more details on these simulations see 5.6.

4.4 Measuring learning parameters

For Figure 4, we compute expected learning speed $\mathbb{E}[\tilde{\nu}_t]$ and local task difficulty $\mathbb{E}[\tilde{G}_t]$ as a function of expansion ratio q (see eq (45) and (50)). We first compute the value of the learning parameters and scaling functions found in the equations (45) and (50).

We train the system described in 4.1 with a weight update given by (9) with $\Delta t_e = 1.0$ and $\Delta t_r = 0.5$. The online gradient ∇L_t^o is computed using local sensitivity analysis. Note that the gradient has to be taken across the system, including the plant, PID controller and cerebellar-like network. Julia is able to compute the gradient across the whole system. Each element of learning rule error $\hat{\epsilon}^{lr}$ is drawn independently from a random normal distribution $\mathcal{N}(0, 1)$.

At each weight update time, we compute the task gradient ∇L_t, γ_t^o, $\nabla \hat{L}_t^{o^T} \mathbf{H}_t \nabla \hat{L}_t^o$ and $(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}$.

We generate expanded networks as described in section 4.2, with expansion ratios $q = 1, 2, \ldots, 11$. In each expanded network, we compute the learning parameters as described above. We approximate the scaling functions $\alpha(q)$, $\beta(q)$ and $\zeta(q)$ fitting the variables with respect to the expansion ratio. We use a fit of the form $f(x) = p_1 x^{p_2}$. In particular, we can get an estimate of ρ, the scaling of the biased curvature.

For Figure 4, we use the scaling functions and the learning parameters found from the simulations to compute the expected learning speed $\mathbb{E}[\tilde{\nu}_t]$ and local task difficulty $\mathbb{E}[\tilde{G}_t]$ as a function of expansion ratio q (see eq 45 and 50). We use these equations to compute the learning speed and local task difficulty as we vary the expansion ratio q and the γ. We test $q = 1, \ldots 20$ and $\gamma \in \frac{1}{q^{0.5}}[0.01, 0.015, 0.002, \ldots, 0.3]$. We normalise the values of ν and G for each q and γ by the value for $q = 1$ and smallest γ.

Acknowledgments

This work was supported by ERC grant No. 716643 FLEXNEURO.

5 Supporting Information

5.1 Expected change in task loss

Consider the system at some time t during the trajectory. The cerebellar weights have a value $\mathbf{w}_t$ and the change in weights from the learning rule is $\delta\mathbf{w}_t$. The change in task loss ΔL_t from the weight update $\delta\mathbf{w}_t$ is

$$\Delta L[\delta\mathbf{w}_t] = L[\mathbf{w}_t + \delta\mathbf{w}_t] - L[\mathbf{w}_t] \tag{63}$$

If the system is learning, i.e. the weight update is beneficial, then $\Delta L[\delta\mathbf{w}_t]$ is negative. A large change in loss at the beginning of learning leads to a large learning speed. The steady state is determined by the loss when the change in task loss becomes zero or negative.

We can use the Taylor expansion on the task loss $L[\mathbf{w}_t + \delta\mathbf{w}_t]$ to rewrite the change in task loss

$$\Delta L[\delta\mathbf{w}_t] = \delta\mathbf{w}_t{}^T \nabla L_t + \frac{1}{2}(\delta\mathbf{w}_t{}^T \mathbf{H}_t \delta\mathbf{w}_t) + O(||\delta\mathbf{w}_t||_2^3) \tag{64}$$

where $\cdot^T$ corresponds to the transpose of a vector and we have used the shortened notation for the task loss gradient $\nabla L_t \equiv \nabla_{\mathbf{w}} L[\mathbf{w}(t)]$, and hessian of the task loss $\mathbf{H}_t \equiv \nabla^2_{\mathbf{w}} L[\mathbf{w}(t)]$. The first term is the contribution from the gradient. It quantifies the correlation between weight update from the learning rule and the perfect gradient descent. The second term quantifies the contribution of the curvature of the task loss surface. In particular, it is proportional to the degree of curvature in the direction of the weight change $\mathbf{w}$. The smaller this term is, the larger loss change and better learning performance. Note that the last term $O(||\delta\mathbf{w}_t||_2^3)$ might not be small, but if we consider small time steps between weight updates then this term will be small compared to the first two. Hence from now on we use the second order approximation of ΔL_t.

Using the weight update equation (9) in the second order approximation of ΔL_t we get

$$\Delta L[\delta\mathbf{w}_t] = -\gamma_t \left[\nabla\hat{L}_t^o{}^T \nabla L_t\right] + \frac{1}{2}\gamma_t^2 \left[\nabla\hat{L}_t^o{}^T \mathbf{H}_t \nabla\hat{L}_t^o\right] \tag{65}$$

$$+ \eta_t^{lr} \left[(\hat{\epsilon}_t^{lr})^T \nabla L_t\right] + \frac{1}{2}(\eta^{lr})^2 \left[(\hat{\epsilon}_t^l r)^T \mathbf{H}_t (\hat{\epsilon}_t^{lr})\right] \tag{66}$$

$$+ \frac{1}{2}\eta^{lr}\gamma_t \left(\nabla\hat{L}_t^o{}^T \mathbf{H}_t \hat{\epsilon}_t^{lr} + (\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \nabla\hat{L}_t^o\right) \tag{67}$$

where $\nabla L_t^o \equiv \nabla_{\mathbf{w}} L^o[t; \mathbf{w}(t)]$ is the shortened notation for the online gradient. The first two terms are the contributions from the task relevant plasticity (i.e. the change in loss in the online gradient direction). The next two terms are the contribution from the learning rule error.

In expectation (over the learning rule error directions), $\hat{\epsilon}_t^{lr}$ is uncorrelated with the derivatives of the task loss ∇L_t and $\mathbf{H}_t$, otherwise it would be a task-relevant term. Hence, on expectation

$$\mathbb{E}[(\hat{\epsilon}_t^{lr})^T \nabla L_t] = 0 \tag{68}$$

$$\mathbb{E}[(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \nabla L_t^o] = \mathbb{E}[\nabla L_t^{oT} \mathbf{H}_t \hat{\epsilon}_t^{lr}] = 0 \tag{69}$$

Hence on expectation the change in loss is as follows

$$\mathbb{E}[\Delta L[\delta\mathbf{w}_t]] = -\gamma_t \left[\nabla\hat{L}_t^o{}^T \nabla L_t\right] + \frac{1}{2}\gamma_t^2 \left[\nabla\hat{L}_t^o{}^T \mathbf{H}_t \nabla\hat{L}_t^o\right] + \frac{1}{2}(\eta_t^{lr})^2 \left[(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}\right] \tag{70}$$

5.2 Hessian projections

We now consider the hessian terms $\nabla\hat{L}_t^o{}^T \mathbf{H}_t \nabla\hat{L}_t^o$ and $(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}$. These terms capture the degree of curvature of the task loss F in the direction of $\nabla\hat{L}_t^o$ and $\hat{\epsilon}_t^{lr}$ at the point in weight space $\mathbf{w}_t$. This depends on the eigenvalues of the Hessian $\mathbf{H}_t$ and how strongly the online gradient $\nabla\hat{L}_t^o$ and the task-irrelevant plasticity $\hat{\epsilon}_t^{lr}$ project onto the different eigenvectors of the Hessian.

We can make this relationship concrete and approximate the dependency of the hessian projection terms on the eigenvalues of the Hessian [54]. First, consider the case where the task loss L is a quadratic function in the weights $\mathbf{w}$. In this case, the Hessian $\mathbf{H}_t$ is constant (i.e. independent of the weights $\mathbf{w}$) and positive definite. Furthermore, a quadratic loss function has a minimum $\mathbf{w}^*$, satisfying $\nabla L[\mathbf{w}^*] = 0$. In this case, the gradient of the loss at any point in weights space $\mathbf{w}_t$ is related to the hessian as follows

$$\nabla L_t = \mathbf{H}_t(\mathbf{w}_t - \mathbf{w}^*) \tag{71}$$

Let $\{\lambda_i, \hat{\mathbf{v}}_i\}, \quad i = 1, \ldots, N$ be the eigenvalue-eigenvector pairs of the Hessian. Start by noting that $\mathbf{H}_t$ is a symmetric matrix: it can then be shown that (i) all its eigenvalues are real numbers; and (ii) eigenvectors associated with different eigenvalues are orthogonal. It is then possible to construct an orthonormal coordinate system from the eigenvectors of the hessian. The vector $\mathbf{w}_t - \mathbf{w}^*$ can be decomposed in the directions of the eigenvectors of the Hessian

$\mathbf{w}_t - \mathbf{w}^* = \sum_{i=1}^N c_i \hat{\mathbf{v}}_i$. Hence the projection of the gradient into the hessian can be written as follows

$$\nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{(\mathbf{w}_t - \mathbf{w}^*)^T \mathbf{H}_t{}^T \mathbf{H}_t \mathbf{H}_t (\mathbf{w}_t - \mathbf{w}^*)}{(\mathbf{w}_t - \mathbf{w}^*)^T \mathbf{H}_t{}^T \mathbf{H}_t (\mathbf{w}_t - \mathbf{w}^*)} \tag{72}$$

$$= \frac{(\sum_{i=1}^N c_i \hat{\mathbf{v}}_i^T) \mathbf{H}_t{}^T \mathbf{H}_t \mathbf{H}_t (\sum_{i=1}^N c_i \hat{\mathbf{v}}_i)}{(\sum_{i=1}^N c_i \hat{\mathbf{v}}_i^T) \mathbf{H}_t{}^T \mathbf{H}_t (\sum_{i=1}^N c_i \hat{\mathbf{v}}_i)} \tag{73}$$

$$= \frac{\sum_{i=1}^N c_i^2 \lambda_i^3}{\sum_{i=1}^N c_i^2 \lambda_i^2} \tag{74}$$

Yet the weight state $\mathbf{w}_t$ at some time t could be anywhere in weight space, hence we expect $\mathbf{w}_t - \mathbf{w}^*$ to project evenly onto the eigenvectors of the Hessian [1]. In that case, all the projection coefficients are equal $c_i = c_j = c \quad \forall i, j = 1, \ldots, N$ and we get

$$\nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{\sum_{i=1}^N c^2 \lambda_i^3}{\sum_{i=1}^N c^2 \lambda_i^2} \tag{75}$$

$$= \frac{\sum_{i=1}^N \lambda_i^3}{\sum_{i=1}^N \lambda_i^2} \tag{76}$$

$$= \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{77}$$

Where in the last line we have used an algebraic property of the relationship of the eigenvalues of a matrix and its trace. This gives us an explicit expression for the projection of the gradient ∇L_t. What about the online gradient ∇L_t^o? In one extreme case, the online gradient is an exact approximation of the task gradient $\nabla L_t^o = \nabla L_t$. In that case, the Hessian projections are equal

$$\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o = \nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{78}$$

The less correlated the online gradient and the task gradient are, the smaller we expect this projection to be. Indeed, the only way the projection would increase is if the online gradient projected more strongly onto the directions of large curvature. Yet that would imply knowledge of the Hessian of the task loss which is unlikely. Hence, we expect

$$\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o \leq \nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{79}$$

We can also bound the online gradient projection from the other side. Indeed, the other extreme case occurs when the online gradient is completely uncorrelated with the Hessian. In that case, it projects evenly onto all the eigenvectors and $\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o = \frac{Tr(\mathbf{H}_t)}{N}$. Combining these two results we obtain the following bound

$$\frac{Tr(\mathbf{H}_t)}{N} \leq \nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o \leq \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{80}$$

This simplified case illustrates the dependency of the Hessian term on spread of the eigenvalues. In our situation, the task loss is a non-linear function of the weights $\mathbf{w}$. However, we still expect the projection of the task gradient, online gradient and online learning error to depend on the distribution of the hessian eigenvalues. In the non-linear case, the Hessian is no longer constant throughout weight space. In general, the task loss will be non-convex. Hence, it might have multiple local minima, maxima and saddle points. Let $\mathbf{w}^*$ be a local minimum in the neighborhood of $\mathbf{w}$. We can write the gradient of the task loss at $\mathbf{w}_t$ as follows

$$\nabla L_t = \mathbf{M}_t (\mathbf{w}_t - \mathbf{w}^*) \tag{81}$$

where

$$\mathbf{M}_t = \int_0^1 \nabla^2 F[\gamma(s)] ds \tag{82}$$

[1] The weights state $\mathbf{w}_t$ is not always unbiased in weights space. The learning rule determines the trajectory in weights space constraining how $\mathbf{w}_t - \mathbf{w}^*$ projects onto the eigenvector of the Hessian (see notebook on "Analysis of Quadratic Functions" python notebook). Perfect gradient descent, for example, leads to a weight trajectory that oscillates from large to small projection onto the Hessian.

and $\gamma(s) = s\mathbf{w}^* + (1-s)\mathbf{w}_t, \quad s \in [0,1]$ is the parametrization of the straight line connecting $\mathbf{w}_t$ and $\mathbf{w}^*$. The vector ∇L_t can be decomposed in the directions of the eigenvectors of the Hessian $\mathbf{M}_t(\mathbf{w}_t - \mathbf{w}^*) = \sum_{i=1}^N d_i \hat{\mathbf{v}}_i$. Hence the hessian projection of the gradient is

$$\nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{\sum_{i=1}^N d_i^2 \lambda_i}{\sum_{i=1}^N d_i^2} \tag{83}$$

We consider two cases. First, close to steady state i.e. when $||\mathbf{w}_t - \mathbf{w}^*||_2$ is small, then $\mathbf{M} \approx \mathbf{H}_t$ and $d_i = c_i \lambda_i$ and the argument from the quadratic task loss applies. The bounds in equation 80 still hold.

Second, as $||\mathbf{w}_t - \mathbf{w}^*||_2$ grows, i.e. further away from steady state, $\mathbf{M}$ is less similar than $\mathbf{H}_t$. If $\mathbf{M}_t$ and $\mathbf{H}_t$ were completely independent, then the projection of the gradient onto the different eigenvectors of the Hessian is approximatively even and

$$\nabla \hat{L}_t{}^T \mathbf{H}_t \nabla \hat{L}_t = \frac{\sum_{i=1}^N d_i^2 \lambda_i}{\sum_{i=1}^N d_i^2} = \frac{Tr(\mathbf{H}_t)}{N} \tag{84}$$

and the bounds in equation 80 still hold.

Hence, for both linear and non-linear task loss functions and both at steady state and far from steady state the bounds in equation 80 apply. We make two further remarks. First, we expect that at the beginning of training the projection $\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o$ is closer to the lower bound $\frac{Tr(\mathbf{H}_t)}{N}$ and close to steady state closer to the upper bound. Second, the better the online gradient approximation is, the closer the hessian projection will be to the upper bound. Third, we don't expect the projection will be equal to the lower bound for long while the task loss is decreasing. If it did, the change in weights would be uncorrelated with the Hessian and would behave like learning rule error.

5.3 Gradient and Hessian derivations

Both the gradient of the task loss and online loss have similar forms

$$\nabla L_t = \frac{1}{T} \int_{t=0}^{T} \left(y(t;\mathbf{w}) - r(t)\right) \nabla_{\mathbf{w}} y(t;\mathbf{w}) dt \tag{85}$$

$$\nabla L_t^o = \frac{1}{\Delta t_e - \Delta t_r} \int_{t-\Delta t_e}^{t-\Delta t_r} \left(y(t';\mathbf{w}) - r(t')\right) \nabla_{\mathbf{w}} y(t';\mathbf{w}) dt' \tag{86}$$

The gradients depend on the gradient of the plant output $\nabla_{\mathbf{w}} y(t;\mathbf{w})$. The output of the system depends on the network weights $\mathbf{w}$ through the plant input $\mathbf{u}$.

$$\begin{aligned}
\nabla_{\mathbf{w}} y(t;\mathbf{w}) &= \nabla_{\mathbf{w}} g(\mathbf{x}(t;\mathbf{w})) && (87)\\
&= \nabla_{\mathbf{x}} g(\mathbf{x}(t;\mathbf{w})) \nabla_{\mathbf{w}} \mathbf{x}(t;\mathbf{w}) && (88)\\
&= \nabla_{\mathbf{x}} g(\mathbf{x}(t;\mathbf{w})) \int_0^t \frac{d}{du} f(\mathbf{x}(t'), u(t';\mathbf{w})) \mathbf{h}(t') + \nabla_{\mathbf{x}} f(\mathbf{x}(t'), u(t';\mathbf{w})) \nabla_{\mathbf{w}} \mathbf{x}(t) dt' && (89)
\end{aligned}$$

Hence the gradient of the task loss is

$$\begin{aligned}
\nabla L_t = \frac{1}{T} \int_0^T (y(t;\mathbf{w}) - r(t)) [\nabla_{\mathbf{x}} g(\mathbf{x}(t;\mathbf{w})) \int_0^t \frac{d}{du} f(\mathbf{x}(t'), u(t';\mathbf{w})) \mathbf{h}(t') && (90)\\
+ \nabla_{\mathbf{x}} f(\mathbf{x}(t'), u(t';\mathbf{w})) \nabla_{\mathbf{w}} \mathbf{x}(t) dt'] dt && (91)
\end{aligned}$$

and the online gradient is

$$\begin{aligned}
\nabla L_t^o = \frac{1}{\Delta t_e - \Delta t_r} \int_{t-\Delta t_e}^{t} (y(t';\mathbf{w}) - r(t')) [\nabla_{\mathbf{x}} g(\mathbf{x}(t;\mathbf{w})) \int_0^t \frac{d}{du} f(\mathbf{x}(t'), u(t';\mathbf{w})) \mathbf{h}(t') && (92)\\
+ \nabla_{\mathbf{x}} f(\mathbf{x}(t'), u(t';\mathbf{w})) \nabla_{\mathbf{w}} \mathbf{x}(t) dt'] dt && (93)
\end{aligned}$$

The hessian

$$\mathbf{H}_t = \nabla^2_{\mathbf{w}} F[\mathbf{w}] \tag{94}$$

$$= \int_0^T \nabla_{\mathbf{w}} \left[(y(t;\mathbf{w}) - r(t)) \nabla_{\mathbf{w}} y(t;\mathbf{w}) \right] dt \tag{95}$$

$$= \int_0^T \nabla_{\mathbf{w}} y(t;\mathbf{w}) \nabla_{\mathbf{w}} y(t;\mathbf{w})^T + (y(t;\mathbf{w}) - r(t)) \nabla^2_{\mathbf{w}} y(t;\mathbf{w}) dt \tag{96}$$

$$= \int_0^T \nabla_{\mathbf{w}} y(t;\mathbf{w}) \nabla_{\mathbf{w}} y(t;\mathbf{w})^T + 0 dt \tag{97}$$

$$= \int_0^T \nabla_{\mathbf{w}} y(t;\mathbf{w}) \nabla_{\mathbf{w}} y(t;\mathbf{w})^T dt \tag{98}$$

$$= \int_0^T \left[\nabla_{\mathbf{x}} g(\mathbf{x}(t;\mathbf{w})) \int_0^t \frac{d}{du} f(\mathbf{x}(t'), u(t';\mathbf{w})) \mathbf{h}(t') + \nabla_{\mathbf{x}} f(\mathbf{x}(t'), u(t';\mathbf{w})) \nabla_{\mathbf{w}} \mathbf{x}(t) dt' \right] \tag{99}$$

$$\left[\nabla_{\mathbf{x}} g(\mathbf{x}(t;\mathbf{w})) \int_0^t \frac{d}{du} f(\mathbf{x}(t'), u(t';\mathbf{w})) \mathbf{h}(t') + \nabla_{\mathbf{x}} f(\mathbf{x}(t'), u(t';\mathbf{w})) \nabla_{\mathbf{w}} \mathbf{x}(t) dt' \right]^T dt \tag{100}$$

$$= \int_0^T \nabla_{\mathbf{x}} g(\mathbf{x}(t;\mathbf{w})) \nabla_{\mathbf{x}} g(\mathbf{x}(t;\mathbf{w}))^T \tag{101}$$

$$\left[\left[\int_0^t \frac{d}{du} f(\mathbf{x}(t'), u(t';\mathbf{w})) \mathbf{h}(t') dt' \right] \left[\int_0^t \frac{d}{du} f(\mathbf{x}(t'), u(t';\mathbf{w})) \mathbf{h}(t') dt' \right]^T \right. \tag{102}$$

$$\left. + \left[\int_0^t \nabla_{\mathbf{x}} f(\mathbf{x}(t'), u(t';\mathbf{w})) \nabla_{\mathbf{w}} \mathbf{x}(t) dt' \right] \left[\int_0^t \nabla_{\mathbf{x}} f(\mathbf{x}(t'), u(t';\mathbf{w})) \nabla_{\mathbf{w}} \mathbf{x}(t) dt' \right]^T \right] dt \tag{103}$$

$$= \int_0^T (\text{plant dynamic terms}) \left[\int_0^t \mathbf{h}(t') dt' \right] \left[\int_0^t \mathbf{h}(t') dt' \right]^T dt \tag{104}$$

5.4 Scaling of hessian projection terms

Lets define the matrix with the granule cell component of the Hessian matrix

$$\mathbf{G}[t] \equiv \left[\int_0^t \mathbf{h}(t') dt' \right] \left[\int_0^t \mathbf{h}(t') dt' \right]^T \tag{105}$$

We assume the effect of the network expansion on the Hessian is mainly due to the change in the granule cell representation $\mathbf{h}$ and the matrix $\mathbf{G}$ and not the change in plant dynamics.

We found that the projection of the online gradient onto the Hessian is bound by $Tr(\mathbf{H}_t)/N$ and $Tr({\mathbf{H}_t}^3)/Tr({\mathbf{H}_t}^2)$ as in equation 27. Without knowing the plant dynamics we can't calculate directly the projections in the initial and expanded network or the Hessian but we can consider how the quantities change relatively with the expansion.

We find that the trace increases linearly

$$Tr(\tilde{\mathbf{G}}) \propto \left(\frac{\tilde{N}}{N} \right) Tr(\mathbf{G}) \tag{106}$$

It follows that the average eigenvalue remains constant with the expansion

$$\frac{Tr(\tilde{\mathbf{G}})}{\tilde{N}} \approx \frac{Tr(\mathbf{G})}{N} \tag{107}$$

Furthermore, as the input expansion increases, the spread of the eigenvalues increases at most linearly.

$$O\left(\frac{Tr(\tilde{\mathbf{G}}^3)}{Tr(\tilde{\mathbf{G}}^2)} \right) = O\left(\tilde{N} \right) \tag{108}$$

As we assume the effect of the network expansion on the Hessian is mainly due to the change in the granule cell representation $\mathbf{h}$ and the matrix $\mathbf{G}$ and not the change in plant dynamics. Then, the same scaling applies to the hessian

terms.

$$\frac{Tr(\tilde{\mathbf{H}}_t)}{\tilde{N}} \approx \frac{Tr(\mathbf{H}_t)}{N} \tag{109}$$

and

$$\frac{Tr(\tilde{\mathbf{H}}_t{}^3)}{Tr(\tilde{\mathbf{H}}_t{}^2)} \leq \left(\frac{\tilde{N}}{N}\right) \frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{110}$$

Using the bounds in equation 27, we get the following bounds for the hessian projection in the lager network.

$$\frac{Tr(\mathbf{H}_t)}{N} \approx \frac{Tr(\tilde{\mathbf{H}}_t)}{\tilde{N}} \leq \nabla\hat{\tilde{L}}_t^{oT}\tilde{\mathbf{H}}_t\nabla\hat{\tilde{L}}_t^o \leq \frac{Tr(\tilde{\mathbf{H}}_t{}^3)}{Tr(\tilde{\mathbf{H}}_t{}^2)} \leq \frac{\tilde{N}}{N}\frac{Tr(\mathbf{H}_t{}^3)}{Tr(\mathbf{H}_t{}^2)} \tag{111}$$

Hence, the projection of the online gradient onto the Hessian scales at most linearly with the network expansion.

$$\nabla\hat{\tilde{L}}_t^{oT}\tilde{\mathbf{H}}_t\nabla\hat{\tilde{L}}_t^o \approx \left(\frac{\tilde{N}}{N}\right)^{\rho} [\nabla\hat{L}_t^{o}{}^T\mathbf{H}_t\nabla\hat{L}_t^o] \tag{112}$$

with $0 \leq \rho \leq 1$ is the coefficient determining the scaling of the hessian projection.

We find that the Hessian projection $\nabla\hat{L}_t^{o}{}^T\mathbf{H}_t\nabla\hat{L}_t^o$ depends on γ_t^o. Indeed, the more correlated the online gradient is with the task gradient the more heavily it projects onto the directions of larger curvature. Similarly, ρ mainly depends on γ_t^o. From simulations, we find the following fit

$$\left(\nabla\hat{\tilde{L}}_t^o\right)^T\mathbf{H}_t\nabla\tilde{\hat{L}}_t^o = (\gamma_t^o)^2 q^{\rho(\gamma_t^o)}\nabla\hat{L}_t^{o}{}^T\mathbf{H}_t\nabla\hat{L}_t^o \tag{113}$$

with

$$\rho(\gamma_t^o) = \left|\frac{1}{(1+e^{-2.5\gamma_t^o})}\right| \tag{114}$$

Hence, the less correlated the online gradient is with the task gradient (smaller γ_t^o, the slower it scales with q. Figure 1 D) shows the fit.

This has an effect on the scaling of learning performance as a function of γ^o. The more less correlated the online gradient is with the task gradient, the more beneficial the input expansion is for learning (see 2). We observe that as γ^o increases, the slower the learning speed increases with q. For larger γ^o, the learning speed initially increases, until it reaches a maximal value, beyond which the learning speed decreases. The larger γ^o is, the larger the local task difficulty is. Indeed, near steady state, the effect of the curvature in the weight update is diminished when the change in weights is less correlated with the task gradient. Furthermore, for smaller γ^o, the local task difficulty decreases more rapidly with q.

5.5 Optimal plasticity magnitude

Given a direction of plasticity $\delta\hat{\mathbf{w}}_t$, given by equation 9, we can compute the optimal plasticity magnitude $||\delta\mathbf{w}_t||$ that leads to the largest decrease in task loss in that direction. We write the change in task loss given by the Taylor expansion up to second order (see eq (64))

$$\Delta L[\delta\mathbf{w}_t] = \delta\mathbf{w}_t{}^T\nabla L_t + \frac{1}{2}(\delta\mathbf{w}_t{}^T\mathbf{H}_t\delta\mathbf{w}_t) \tag{115}$$

$$= ||\delta\mathbf{w}_t||\delta\hat{\mathbf{w}}_t{}^T\nabla L_t + \frac{1}{2}||\delta\mathbf{w}_t||^2(\delta\hat{\mathbf{w}}_t{}^T\mathbf{H}_t\delta\hat{\mathbf{w}}_t) \tag{116}$$

Taking the derivative with respect to $||\delta\mathbf{w}_t||$

$$\frac{d}{d||\delta\mathbf{w}_t||}\Delta L[\delta\mathbf{w}_t] = \delta\hat{\mathbf{w}}_t{}^T\nabla L_t + ||\delta\mathbf{w}_t||(\delta\hat{\mathbf{w}}_t{}^T\mathbf{H}_t\delta\hat{\mathbf{w}}_t) \tag{117}$$

Yet if the system is learning, then $\delta\hat{\mathbf{w}}_t{}^T\nabla\hat{L}_t < 0$. The Hessian projection $\delta\hat{\mathbf{w}}_t{}^T\mathbf{H}_t\delta\hat{\mathbf{w}}_t$ depends on the curvature of the task loss surface in the neighborhood of the point $\mathbf{w}_t$. In particular, if the surface has some directions of negative curvature (i.e. downward), then the Hessian is not positive definite and the Hessian projection could be negative (if

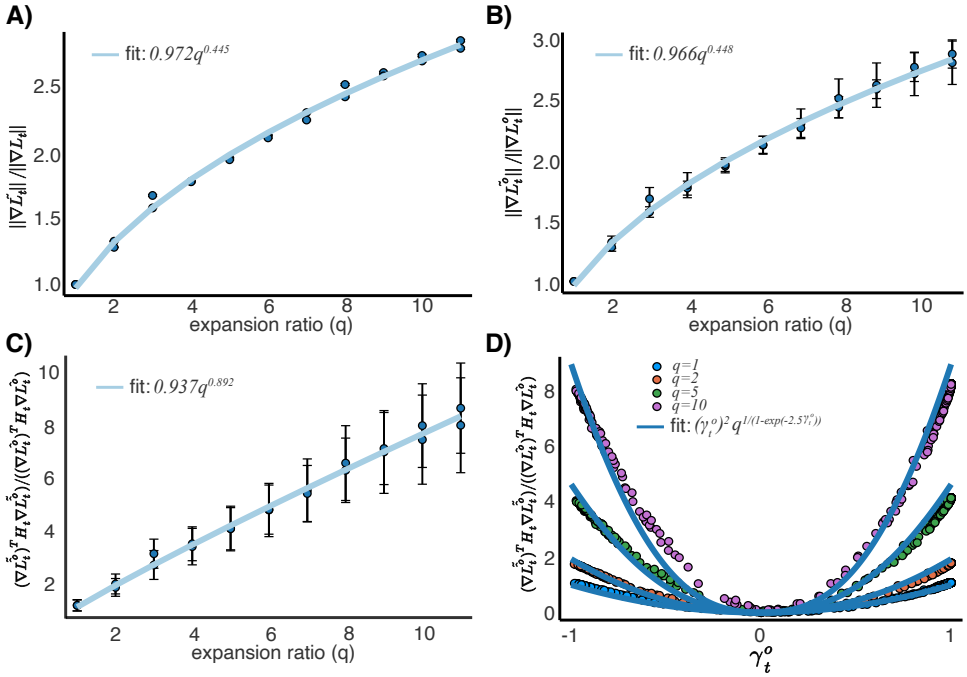

Figure 1: Scaling of learning parameters. **A)**, **B)**, **C)** We plot three different learning parameters for a range of expansion ratios $q = 1, \ldots, 11$, for two different simulations with different weight initialisations and reference trajectories. We fit the data points with a power curve $y = ax^b$. We recover the scaling parameters found in 2.5. **D)** We plot the normalised Hessian projection as a function of γ_t^o for different expansion ratios. Each color dot represents a different q. We plot the fit given by equations (113) and (114)

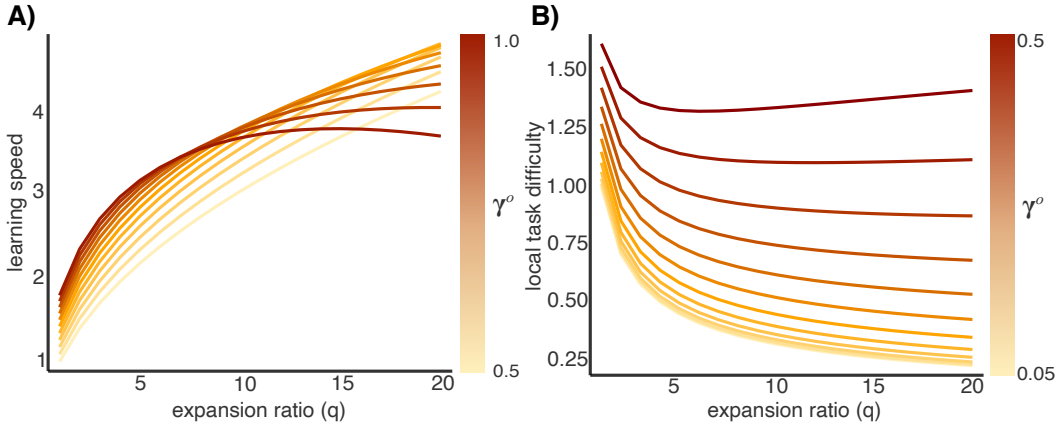

Figure 2: Effect of γ^o on learning speed **A)** and local task difficulty **B)**. We plot the learning performance parameter as a function of the expansion ratio q, for different γ^o. The color of the line represents γ^o.

$\delta\hat{\mathbf{w}}_t$ projects mainly onto the directions of negative curvature). In this case, the change in task loss $\Delta L[\mathbf{w}]$ decreases (becomes more negative) for larger $||\delta \mathbf{w}_t||$ and there is no optimal plasticity magnitude.

If $\delta \mathbf{w}_t$ projects more strongly onto directions of positive curvature (which always happens close to a local minimum), then $\delta\hat{\mathbf{w}}_t{}^T \mathbf{H}_t \delta\hat{\mathbf{w}}_t > 0$. In this case, there is an optimal plasticity magnitude given by the zero of the derivative $\frac{d}{d||\delta \mathbf{w}_t||}\Delta L[\delta \mathbf{w}_t] = 0$. Hence

$$||\delta \mathbf{w}_t||^* = \frac{-\delta\hat{\mathbf{w}}_t{}^T \nabla \hat{L}_t}{\delta\hat{\mathbf{w}}_t{}^T \mathbf{H}_t \delta\hat{\mathbf{w}}_t} ||\nabla L_t|| \tag{118}$$

We consider a online learning rule with the weight plasticity given by equation (9), with a fixed σ. We can find the optimal γ_t^* satisfying $||\delta \mathbf{w}_t||^* = \gamma_t^*(-\nabla \hat{L}_t^o + \frac{1}{\sigma}\hat{\epsilon}_t^{lr})$. Using this with (118) we get

$$\gamma_t^* = \frac{\gamma_t^o ||\nabla L_t||}{\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o + \frac{1}{\sigma^2}(\hat{\epsilon}_t^{lr})^T \mathbf{H}_t \hat{\epsilon}_t^{lr}} \tag{119}$$

If the signal to noise ratio σ is large, then the second term in the denominator is small and we get $\gamma_t^* \approx \frac{\gamma_t^o ||\nabla L_t||}{\nabla \hat{L}_t^o{}^T \mathbf{H}_t \nabla \hat{L}_t^o}$. This is the value of the γ that maximally reduces the task loss. The learning rule error term decreases γ_t^*.

5.6 Notes on simulations

We perform simulations of learning in the motor control system to test our theoretical results. We have three sources of variability in our simulations.

- from the reference trajectory $r(t)$. The reference trajectory for a given simulation is drawn from a family of random reference trajectories as described in 4.1. The variation comes from the selection of the phase shifts ϕ_k

$$r(t) = \frac{c_0}{2} + \sum_{k=1}^{N_{ref}/2} c_k \sin(2\pi f_k t + \phi_k) \tag{120}$$

  with

$$c_k = \frac{c\tau}{2} \frac{2\tau}{1 + (2\pi\tau k)^2} \tag{121}$$

  $r(t)$ influences the task loss $L[\mathbf{w}]$. We expect relatively little variation in the geometry of the task loss as a function of reference trajectory. More importantly, we expect the effect of the network expansion to be more significant than the variation from $r(t)$. Indeed, each reference trajectory is chosen from the same family of trajectories (fourier series with the same cut-off frequency).

- from the weight initialization of both the input weights Z and the output weights $\mathbf{w}$. This weights are initialised for each simulation according to 4.1 for the smallest network and then expanded for the larger networks according to 4.2. The value of Z affects directly the task loss L as it determines the mapping from the mossy fibre inputs to the granule cell layer activity h. Note that Z are constant during the whole trajectory as learning only changes the output weights. This value of the initial output weights $\mathbf{w}_0$ determines the initial point in the loss landscape, which determines both the initial task loss value $L[\mathbf{w}_0]$ and the geometry around that point. We know that the optimal γ^* depends strongly on the geometry (slope and curvature) at the weight point. Hence this initialization will have a big effect on the increase of learning speed and decrease of steady state loss with respect to learning step γ. Furthermore, it will also determine the scaling of the learning parameters (gradient hessian projections) on the expansion ratio. The qualitative evolution for each weight initialization will be the same but the exact values will vary.

- from the learning rule error $\hat{\epsilon}_t^{lr}$ in the weight update rule (5).

$$\delta \mathbf{w}_t = -\gamma e_{\Delta t_e}(t) \mathbf{h}(t - \Delta t_h) + \eta \hat{\epsilon}_t \tag{122}$$

  the vector $\hat{\epsilon}_t$ is drawn from a random distribution (see 4.1). We are interested in the expected learning speed and steady state loss over the learning rule error $\mathbb{E}[\nu]$ and $\mathbb{E}[\xi]$. Hence we should train multiple times starting with the same initializations but with different learning rule error during training. This will lead to different weight trajectories and task loss during learning. We expect some variability in the learning speed and the steady state loss. The variability should be weaker than the change in learning performance from the change in expansion ratio.

In Figure 2 C) D) each dot is the mean over different simulations with different $r(t)$ and weight initialisations. This guarantees that our results are not dependent on the choice of reference trajectory or weight initilisation. It just requires being in a regime in which the system is learning (decreasing the task loss). The scatter plot B) showing the trade-off between learning speed and steady state loss is plotted for a single $r(t)$ and weight initialisation but for 20 different training instances. As mentioned above the weight initialisation can have a strong effect on the geometry of the task loss and hence on the dependence of ν and ξ on γ and the scaling with respect to q. The results of the trade-off and will still hold but the exact factors will vary.

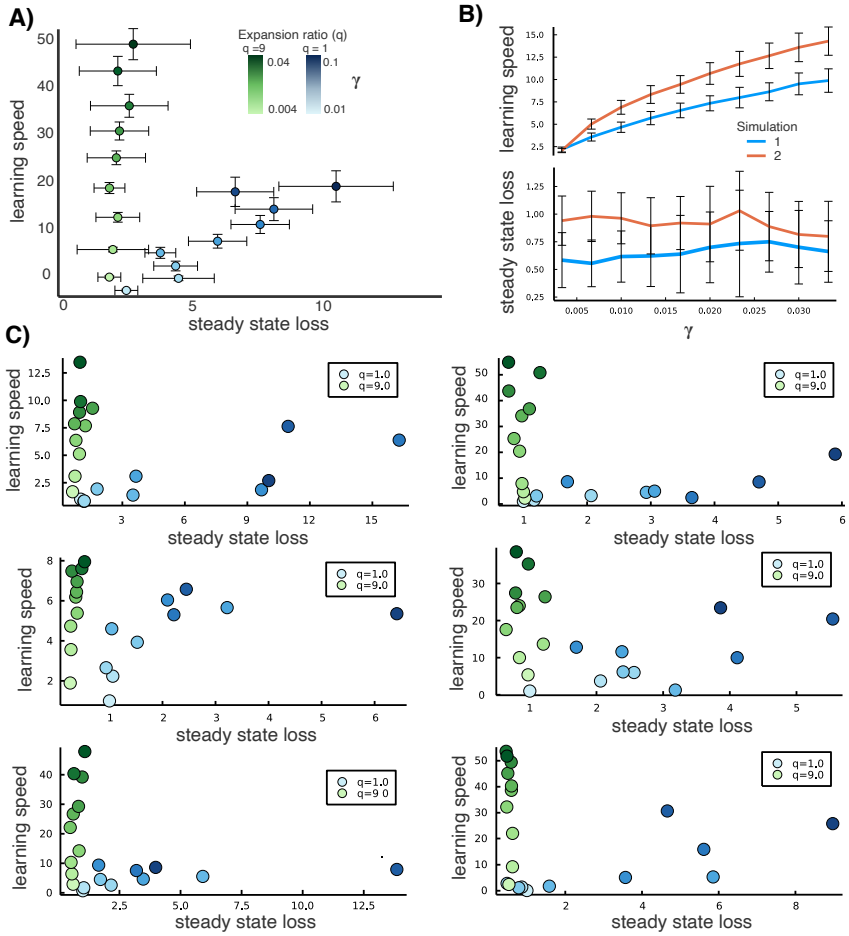

Figure 3: Effect of variability in simulations. **A)** Trade-off between steady state loss and learning speed averaged over 17 different reference trajectories and weight initilisations. Same data as in Figure 2 C) D). The variability is high. **B)** Learning speed and steady state loss as a function of γ for two different simulations. For both simulations the trend in the learning variables is the same but coefficients are different. **C)** Trade-off of steady state loss and learning speed for a subset of 6 different simulations used for **A)**. For each different simulation the trade-off is visible but has different values.

References

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

[2] B. Babadi and H. Sompolinsky. Sparseness and expansion in sensory representations. *Neuron*, 83:1213–1226, 9 2014.

[3] A. Badura and C. I. De Zeeuw. Cerebellar Granule Cells: Dense, Rich and Evolving Representations, jun 2017.

[4] H. Baillieux, H. J. D. Smet, P. F. Paquier, P. P. D. Deyn, and P. Mariën. Cerebellar neurocognition: insights into the bottom of the brain. *Clinical neurology and neurosurgery*, 110:763–773, 8 2008.

[5] J. S. Baizer, I. Kralj-Hans, and M. Glickstein. Cerebellar lesions and prism adaptation in macaque monkeys. *Journal of neurophysiology*, 81:1960–1965, 1999.

[6] A. J. Bastian. Moving, sensing and learning with cerebellar damage. *Current Opinion in Neurobiology*, 21:596–601, 8 2011.

[7] A. J. Bastian, T. A. Martin, J. G. Keating, and W. T. Thach. Cerebellar ataxia: Abnormal control of interaction torques across multiple joints. *Journal of Neurophysiology*, 76(1):492–509, 1996.

[8] G. Billings, E. Piasini, A. Lorincz, Z. Nusser, and R. A. Silver. Network Structure within the Cerebellar Input Layer Enables Lossless Sparse Encoding. *Neuron*, 83(4):960–974, aug 2014.

[9] L. Bottou. Stochastic Gradient Descent Tricks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTURE NO:421–436, 2012.

[10] J. M. Bower. The organization of cerebellar cortical circuitry revisited. *Annals of the New York Academy of Sciences*, 978:135–155, 12 2002.

[11] B. D. Cameron, C. de la Malla, and J. López-Moliner. The role of differential delays in integrating transient visual and proprioceptive information. *Frontiers in Psychology*, 5:50, 2014.

[12] M. R. Carey. Synaptic mechanisms of sensorimotor learning in the cerebellum, 8 2011.

[13] R. R. Carrillo, E. Ros, C. Boucheny, and O. J. Coenen. A real-time spiking cerebellum model for learning robot control. *BioSystems*, 94(1-2):18–27, oct 2008.

[14] C. Casellato, A. Antonietti, J. A. Garrido, R. R. Carrillo, N. R. Luque, E. Ros, A. Pedrocchi, and E. D'Angelo. Adaptive Robotic Control Driven by a Versatile Spiking Cerebellar Network. *PLoS ONE*, 9(11):e112265, nov 2014.

[15] N. A. Cayco-Gajic, C. Clopath, and R. A. Silver. Sparse synaptic connectivity is required for decorrelation and pattern separation in feedforward networks. *Nature Communications*, 8(1), dec 2017.

[16] N. A. Cayco-Gajic and R. A. Silver. Re-evaluating Circuit Mechanisms Underlying Pattern Separation. *Neuron*, 101(4):584–602, 2019.

[17] S. E. Criscimagna-Hemminger, A. J. Bastian, and R. Shadmehr. Size of error affects cerebellar contributions to motor learning. *Journal of Neurophysiology*, 103(4):2275–2284, apr 2010.

[18] A. Diamond. Close interrelation of motor development and cognitive development and of the cerebellum and prefrontal cortex. *Child development*, 71:44–56, 2000.

[19] S. Dieudonné. Submillisecond kinetics and low efficacy of parallel fibre-golgi cell synaptic currents in the rat cerebellum. *The Journal of Physiology*, 510:845, 8 1998.

[20] K. Doya. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks*, 12(7-8):961–974, oct 1999.

[21] J. C. Eccles. The Cerebellum as a Computer: Patterns in Space and Time. *J. Phygiol*, 229:1–32, 1973.

[22] J. I. Gilmer and A. L. Person. Theoretically Sparse, Empirically Dense: New Views on Cerebellar Granule Cells, dec 2018.

[23] A. Giovannucci, A. Badura, B. Deverett, F. Najafi, T. D. Pereira, Z. Gao, I. Ozden, A. D. Kloth, E. Pnevmatikakis, L. Paninski, C. I. De Zeeuw, J. F. Medina, and S. S. Wang. Cerebellar granule cells acquire a widespread predictive feedback signal during motor learning. *Nature Neuroscience*, 20(5):727–734, may 2017.

[24] D. J. Herzfeld, Y. Kojima, R. Soetedjo, and R. Shadmehr. Encoding of error and learning to correct that error by the Purkinje cells of the cerebellum. *Nature Neuroscience*, 21(5):736–743, may 2018.

[25] T. Hirano. Depression and potentiation of the synaptic transmission between a granule cell and a Purkinje cell in rat cerebellar culture. *Neuroscience Letters*, 119(2):141–144, nov 1990.

[26] M. Ito. Movement and thought: identical control mechanisms by the cerebellum. *Trends in Neurosciences*, 16:448–450, 11 1993.

[27] M. Ito and M. Kano. Long-lasting depression of parallel fiber-Purkinje cell transmission induced by conjunctive stimulation of parallel fibers and climbing fibers in the cerebellar cortex. *Neuroscience Letters*, 33(3):253–258, dec 1982.

[28] M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 7 1992.

[29] H. Jörntell, F. Bengtsson, M. Schonewille, and C. I. D. Zeeuw. Cerebellar molecular layer interneurons – computational properties and roles in learning. *Trends in Neurosciences*, 33:524–532, 11 2010.

[30] H. Jörntell and C. F. Ekerot. Properties of somatosensory synaptic integration in cerebellar granule cells in vivo. *Journal of Neuroscience*, 26:11786–11797, 11 2006.

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

[32] M. Kawato and H. Gomi. A computational model of four regions of the cerebellum based on feedback-error learning. *Biological Cybernetics*, 68(2):95–103, dec 1992.

[33] M. Kawato, S. Ohmae, H. Hoang, and T. Sanger. 50 Years Since the Marr, Ito, and Albus Models of the Cerebellum, jun 2020.

[34] S. Kitazawa, T. Kimura, and P. B. Yin. Cerebellar complex spikes encode both destinations and errors in arm movements. *Nature 1998 392:6675*, 392(6675):494–497, apr 1998.

[35] L. D. Knogler, D. A. Markov, E. I. Dragomir, V. Štih, and R. Portugues. Sensorimotor Representations in Cerebellar Granule Cells in Larval Zebrafish Are Dense, Spatially Organized, and Non-temporally Patterned. *Current Biology*, 27(9):1288–1302, may 2017.

[36] M. Küper, A. Dimitrova, M. Thürling, S. Maderwald, J. Roths, H. G. Elles, E. R. Gizewski, M. E. Ladd, J. Diedrichsen, and D. Timmann. Evidence for a motor and a non-motor domain in the human dentate nucleus — an fmri study. 2011.

[37] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient BackProp. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTURE NO:9–48, 2012.

[38] A. Litwin-Kumar, K. D. Harris, R. Axel, H. Sompolinsky, and L. F. Abbott. Optimal Degrees of Synaptic Connectivity. *Neuron*, 2017.

[39] R. R. Llinás, K. D. Walton, and E. J. Lang. Cerebellum. *The Synaptic Organization of the Brain*, 1 2004.

[40] D. Marr. A theory of cerebellar cortex. *The Journal of Physiology*, 202(2):437–470, jun 1969.

[41] M. Maschke, C. M. Gomez, T. J. Ebner, and J. Konczak. Hereditary cerebellar ataxia progressively impairs force adaptation during goal-directed arm movements. *Journal of Neurophysiology*, 91:230–238, 1 2004.

[42] D. Menghini, M. Di Paola, R. Murri, F. Costanzo, C. Caltagirone, S. Vicari, and L. Petrosini. Cerebellar vermis abnormalities and cognitive functions in individuals with Williams syndrome. *Research in Developmental Disabilities*, 34(7):2118–2126, jul 2013.

[43] R. C. Miall, D. J. Weir, D. M. Wolpert, and J. F. Stein. Is the Cerebellum a Smith Predictor? Technical Report 3, 1993.

[44] R. C. Miall and D. M. Wolpert. Forward models for physiological motor control. *Neural Networks*, 9:1265–1279, 1996.

[45] F. A. Middleton and P. L. Strick. Anatomical evidence for cerebellar and basal ganglia involvement in higher cognitive function. *Science*, 266:458–461, 10 1994.

[46] M. Molinari, F. R. Chiricozzi, S. Clausi, A. M. Tedesco, M. D. Lisa, and M. G. Leggio. Cerebellum and detection of sequences, from perception to cognition. *Cerebellum*, pages 611–615, 2008.

[47] S. M. Morton and A. J. Bastian. Cerebellar contributions to locomotor adaptations during splitbelt treadmill walking. *Journal of Neuroscience*, 26:9107–9116, 9 2006. We demonstrated that cerebellar damage does not impair the ability to make reactive feedback-driven adaptations, but significantly disrupts predictive feedforward motor adaptations during splitbelt treadmill locomotion.

[48] F. V. Overwalle, M. Manto, Z. Cattaneo, S. Clausi, C. Ferrari, J. D. Gabrieli, X. Guell, E. Heleven, M. Lupo, Q. Ma, M. Michelutti, G. Olivito, M. Pu, L. C. Rice, J. D. Schmahmann, L. Siciliano, A. A. Sokolov, C. J. Stoodley, K. van Dun, L. Vandervert, and M. Leggio. Consensus paper: Cerebellum and social cognition. *Cerebellum*, 19:833–868, 12 2020.

[49] M. Paulin. A kalman filter theory of the cerebellum, 1989.

[50] L. S. Popa and T. J. Ebner. Cerebellum and internal models. *Handbook of the Cerebellum and Cerebellar Disorders*, pages 1–25, 2013.

[51] J. Porrill, P. Dean, and S. R. Anderson. Adaptive filters and internal models: Multilevel description of cerebellar function. *Neural Networks*, 47:134–149, 11 2013.

[52] K. Rabe, O. Livne, E. R. Gizewski, V. Aurich, A. Beck, D. Timmann, and O. Donchin. Adaptation to visuomotor rotation and force field perturbation is correlated to different brain areas in patients with cerebellar degeneration. *Journal of Neurophysiology*, 101:1961–1971, 4 2009.

[53] D. V. Raman and T. O'Leary. Frozen algorithms: how the brain's wiring facilitates learning. *Current Opinion in Neurobiology*, 67:207–214, 4 2021.

[54] D. V. Raman and T. O'leary. Optimal plasticity for memory maintenance during ongoing synaptic change. *eLife*, 10, sep 2021.

[55] D. V. Raman, A. P. Rotondo, and T. O'Leary. Fundamental bounds on learning performance in neural circuits. *Proceedings of the National Academy of Sciences of the United States of America*, 116(21):10537–10546, may 2019.

[56] J. L. Raymond and J. F. Medina. Computational Principles of Supervised Learning in the Cerebellum. *Annual Review of Neuroscience*, 41(1):233–253, jul 2018.

[57] T. D. Sanger, O. Yamashita, and M. Kawato. Expansion coding and computation in the cerebellum: 50 years after the Marr–Albus codon theory. *The Journal of Physiology*, 598(5):913–928, mar 2020.

[58] J. A. Saunders and D. C. Knill. Visual feedback control of hand movements. *Journal of Neuroscience*, 24:3223–3234, 3 2004.

[59] J. D. Schmahmann. The cerebellum and cognition. *Neuroscience letters*, 688:62–75, 1 2019.

[60] N. Schweighofer, K. Doya, and F. Lay. Unsupervised learning of granule cell sparse codes enhances cerebellar adaptive control. *Neuroscience*, 103:35–50, 2 2001.

[61] N. T. Silva, J. Ramírez-Buriticá, D. L. Pritchett, and M. R. Carey. Neural instructive signals for associative cerebellar learning. *bioRxiv*, page 2022.04.18.488634, 4 2022.

[62] M. A. Smith, J. Brandt, and R. Shadmehr. Motor Disorder in Huntington's Disease Begins as a Dysfunction in Error Feedback Control. *Nature*, 403(6769):544, feb 2000.

[63] M. A. Smith and R. Shadmehr. Intact ability to learn internal models of arm dynamics in Huntington's disease but not cerebellar degeneration. *Journal of Neurophysiology*, 93(5):2809–2821, may 2005.

[64] A. Spanne and H. Jörntell. Questioning the role of sparse coding in the brain. *Trends in Neurosciences*, 38:417–427, 7 2015.

[65] C. J. Stoodley, E. M. Valera, and J. D. Schmahmann. Functional topography of the cerebellum for motor and cognitive tasks: An fMRI study. *NeuroImage*, 59(2):1560–1570, jan 2012.

[66] M. L. Streng, L. S. Popa, and T. J. Ebner. Climbing fibers control purkinje cell representations of behavior. *Journal of Neuroscience*, 37(8):1997–2009, feb 2017.

[67] A. Suvrathan, H. L. Payne, and J. L. Raymond. Timing Rules for Synaptic Plasticity Matched to Behavioral Function. *Neuron*, 92(5):959–967, dec 2016.

[68] T. Torioka. Pattern separability and the effect of the number of connections in a random neural net with inhibitory connections, 1978.

[69] T. Tyrrell and D. Willshaw. Cerebellar cortex: its simulation and the relevance of marr's theory. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 336:239–257, 5 1992.

[70] A. Vijayan, V. Gopan, B. Nair, and S. Diwakar. Comparing robotic control using a spiking model of Cerebellar network and a gain adapting forward-inverse model. In *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, volume 2017-January, pages 566–570. Institute of Electrical and Electronics Engineers Inc., nov 2017.

[71] M. J. Wagner, T. H. Kim, J. Savall, M. J. Schnitzer, and L. Luo. Cerebellar granule cells encode the expectation of reward. *Nature*, 544(7648):96–100, apr 2017.

[72] B. Widrow and M. Hoff. Adaptive switching circuits. 1960.

[73] D. M. Wolpert, R. C. Miall, and M. Kawato. Internal models in the cerebellum, sep 1998.