Google Research

Learning at Multiple Timescales

Matt Jones, mcjones@google.com

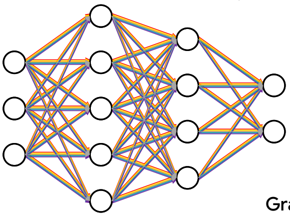
MemARI
NeurIPS 2022 | December 2nd
New Orleans, USA

Contributions

- General framework for learning/memory in nonstationary domains
  - Parallel systems with different characteristic timescales
- Multiscale Optimizer (NN implementation)
  - Subweights with different learning and decay rates
- Model equivalence results
  - Eliminate extraneous coupling between timescales
- New perspective on momentum
  - Equivalent to fast weight with negative learning rate

Multiscale Optimizer

Decompose each weight w_j as a sum of subweights ω_{ij}



$\boldsymbol{\omega}_n$ slow: small α, $\gamma \approx 1$
$\boldsymbol{\omega}$.
$\boldsymbol{\omega}$.
$\boldsymbol{\omega}$.
$\boldsymbol{\omega}_2$.
$\boldsymbol{\omega}_1$ fast: large α, small γ

$$w_j = \sum_{i=1}^{n} \omega_{ij}$$

Timescale i
Weight j

Gradient descent with decay:

$$\omega_{ij}(t+1) = \underbrace{\gamma_i}_{\text{decay factor}}\, \omega_{ij}(t) - \underbrace{\alpha_i}_{\text{learning rate}}\, \underbrace{\partial_{w_j}\mathcal{L}(t)}_{\text{loss gradient}}$$

Model Equivalence: Eliminating Coupling between Timescales

Benna-Fusi Model Synapse [BF16]

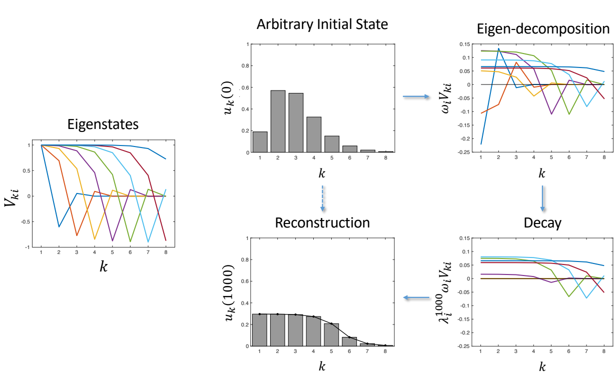
- Adopted in continual reinforcement learning [KSC18, KSC19]
- Coupled biochemical processes at different timescales

$$C_1 \Delta u_1 = g_1(u_2 - u_1) - \alpha \partial_w \mathcal{L}$$
$$C_k \Delta u_k = g_{k-1}(u_{k-1} - u_k) + g_k(u_{k+1} - u_k)$$

$$w = u_1 \leftrightarrow u_2 \leftrightarrow u_3 \leftrightarrow u_4 \leftrightarrow u_5 \leftrightarrow u_6 \leftrightarrow u_7$$

Reparameterization

- Linear dynamics: $\boldsymbol{u}(t+1) = \boldsymbol{T}\boldsymbol{u}(t) + \boldsymbol{d}(t)$
- Eigenvector coordinate change: $\boldsymbol{T} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$, $\boldsymbol{\omega} := \boldsymbol{V}^{-1}\boldsymbol{u}$
- Simplified dynamics: $\boldsymbol{\omega}(t+1) = \boldsymbol{\Lambda}\boldsymbol{\omega}(t) + \boldsymbol{V}^{-1}\boldsymbol{d}(t)$
- Instance of multiscale optimizer: $\omega_i(t+1) = \lambda_i \omega_i(t) + \alpha_i(t)$

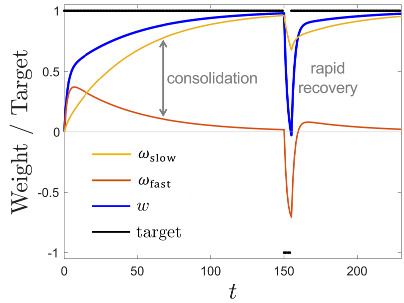


Fast Weights

- Contrast with fast weights in recurrent networks [BHM+16, HP87]
- Special case of multiscale optimizer

$$\boldsymbol{\omega}_{\text{slow}}(t+1) = \boldsymbol{\omega}_{\text{slow}}(t) + \alpha_{\text{slow}} \partial_{\boldsymbol{w}} \mathcal{L}(t)$$
$$\boldsymbol{\omega}_{\text{fast}}(t+1) = \gamma_{\text{fast}} \boldsymbol{\omega}_{\text{fast}}(t) + \alpha_{\text{fast}} \partial_{\boldsymbol{w}} \mathcal{L}(t)$$

- Fast weights adapt quickly, protects from catastrophic forgetting



Momentum as Negative Fast Weight

- Standard momentum learning [RHW86, Qia99]

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \eta \boldsymbol{m}(t+1)$$
$$\boldsymbol{m}(t+1) = \beta \boldsymbol{m}(t) + (1-\beta) \partial_{\boldsymbol{w}} \mathcal{L}(t)$$

- Same eigenvector trick:

$$\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{m} \end{bmatrix}_{(t+1)} = \underbrace{\begin{bmatrix} 1 & -\eta\beta \\ 0 & \beta \end{bmatrix}}_{} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{m} \end{bmatrix}_{(t)} - \begin{bmatrix} \eta(1-\beta) \\ -(1-\beta) \end{bmatrix} \partial_{\boldsymbol{w}} \mathcal{L}(t)$$

$$= \begin{bmatrix} 1 & \frac{1}{\frac{1-\beta}{\eta\beta}} \\ 0 & \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & \frac{1-\beta}{\eta\beta} \end{bmatrix}^{-1}$$

$$\begin{bmatrix} \boldsymbol{\omega}_{\text{slow}} \\ \boldsymbol{\omega}_{\text{fast}} \end{bmatrix} := \begin{bmatrix} 1 & 1 \\ 0 & \frac{1-\beta}{\eta\beta} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{m} \end{bmatrix}$$

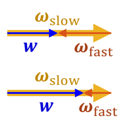
- Multiscale optimizer with negative fast learning rate

$$\begin{bmatrix} \boldsymbol{\omega}_{\text{slow}} \\ \boldsymbol{\omega}_{\text{fast}} \end{bmatrix}_{(t+1)} = \begin{bmatrix} 1 & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} \boldsymbol{\omega}_{\text{slow}} \\ \boldsymbol{\omega}_{\text{fast}} \end{bmatrix}_{(t)} - \begin{bmatrix} \eta \\ -\eta\beta \end{bmatrix} \partial_{\boldsymbol{w}} \mathcal{L}(t)$$

- Explanation: decay of $\boldsymbol{\omega}_{\text{fast}}$ leads $\boldsymbol{w}$ to continue learning toward $\boldsymbol{\omega}_{\text{slow}}$

Initial update:

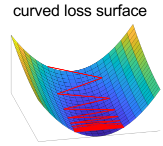
After decay:



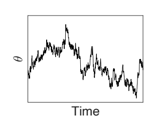
$$\boldsymbol{w} = \boldsymbol{\omega}_{\text{slow}} + \boldsymbol{\omega}_{\text{fast}}$$

- Opposing rationales of momentum and fast weights
  - Momentum: smooths endogenous negative autocorrelation
  - Fast weights: leverage exogenous positive autocorrelation [JSE+22]

Gradient descent on curved loss surface



Nonstationarity in latent task parameters



References

[BF16] Marcus K. Benna and Stefano Fusi. Computational principles of synaptic memory consolidation. *Nature Neuroscience*, 19, 2016.

[BHM+16] Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.

[HP87] Geoffrey E. Hinton and David C. Plaut. Using fast weights to deblur old memories. In *Proceedings of the 9th annual conference of the cognitive science society*, pages 177–186, 1987.

[JSE+22] Matt Jones, Tyler Scott, Gamaleldin ElSayed, Mengye Ren, Katherine Hermann, David Mayo, and Michael C. Mozer. Neural network online training with sensitivity to multiscale temporal structure. In *NeurIPS workshop on Memory in Artificial and Real Intelligence (MemARI)*, 2022.

[KSC18] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018. PMLR 80.

[KSC19] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, 2019. PMLR 97.

[Qia99] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel distributed processing, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA, 1986.