Efficient and robust optimisation of models of protein conformational dynamics

Rob Arbon, rob@redesignscience.com,
30th August 2022, SCOTChem

Conformational dynamics

Conformational dynamics are ubiquitous in biology

- Enzyme catalysis
- Cell signalling
- Drug activity

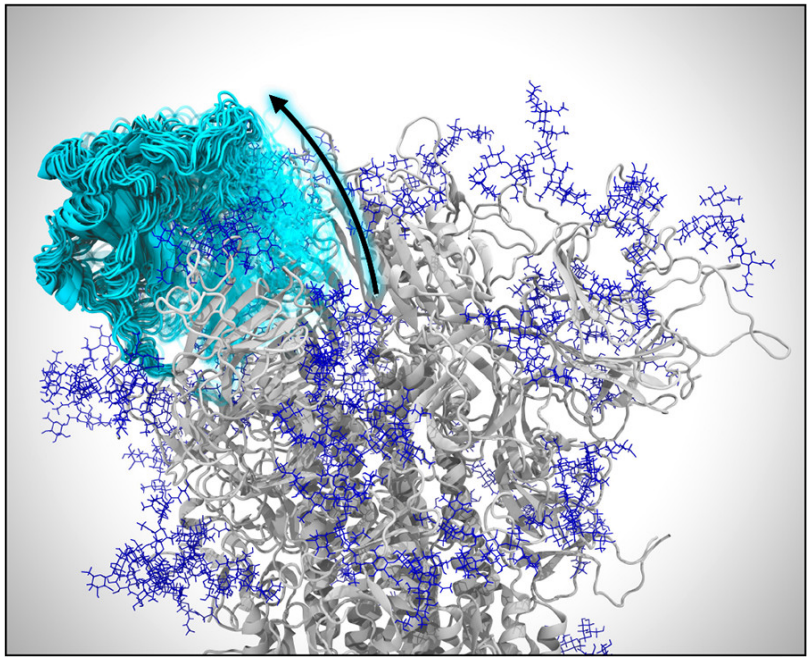


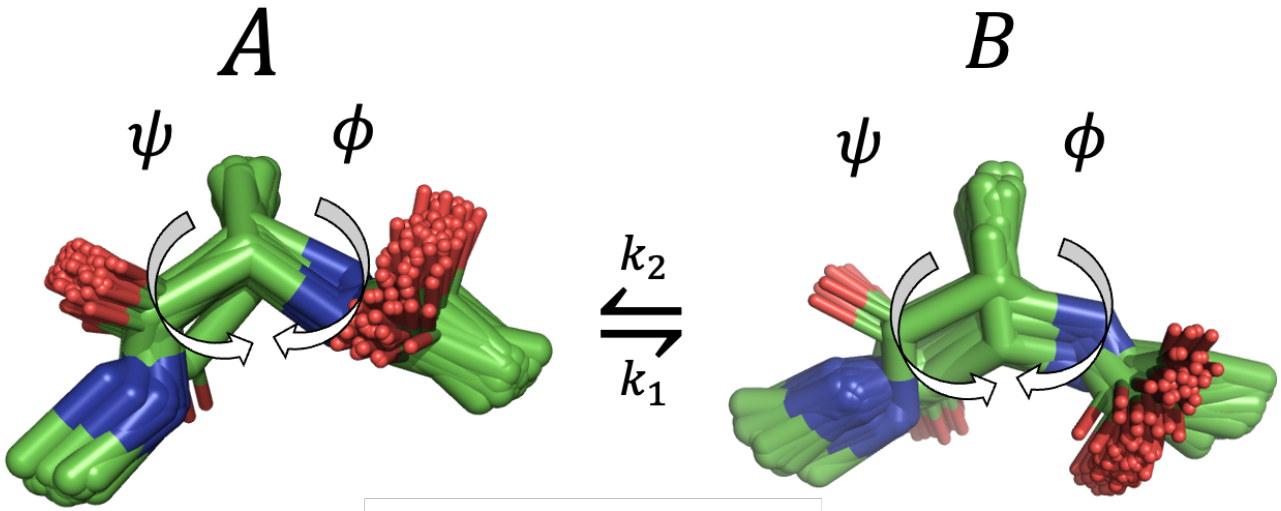
Figure 2. Opening of the spike protein. VMD visualization of weighted ensemble simulations shows the transition of the spike's RBD from the closed state to the open state. Many conformations of the RBD along its opening pathway are represented at the same time using cyan cartoons and a transparency gradient. Glycans appear as dark blue.

Casalino L, Dommer AC, Gaieb Z, et al. AI-driven multiscale simulations illuminate mechanisms of SARS-CoV-2 spike dynamics. The International Journal of High Performance Computing Applications. 2021;35(5):432-451. doi:10.1177/10943420211006452

What are conformational dynamics?

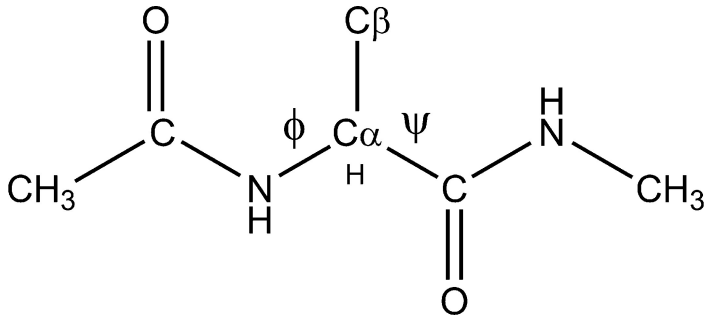
1. Assign structures to A, B, …, (metastable)

2. Determine of k_1, k_2, …



$$\frac{d[A]}{dt} = -k_1[A] + k_2[B]$$

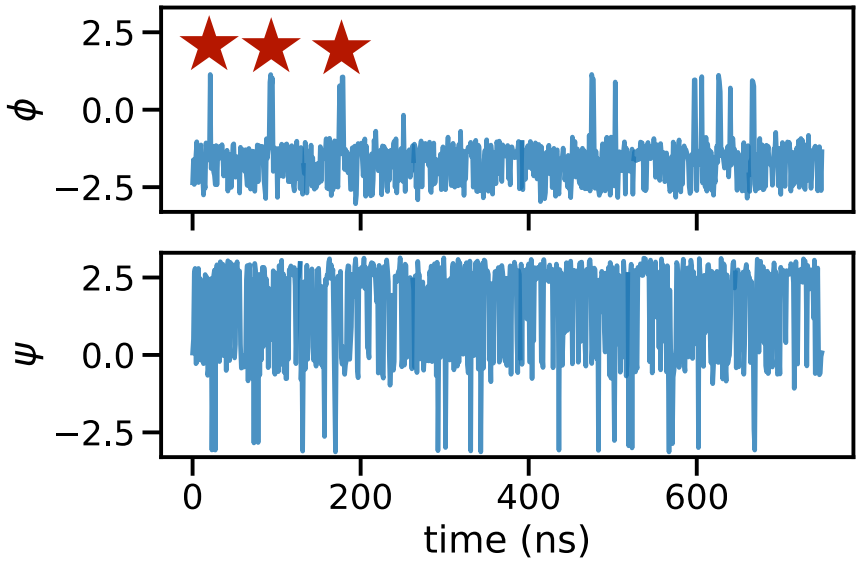
$$\frac{d[B]}{dt} = -k_2[B] + k_1[A]$$

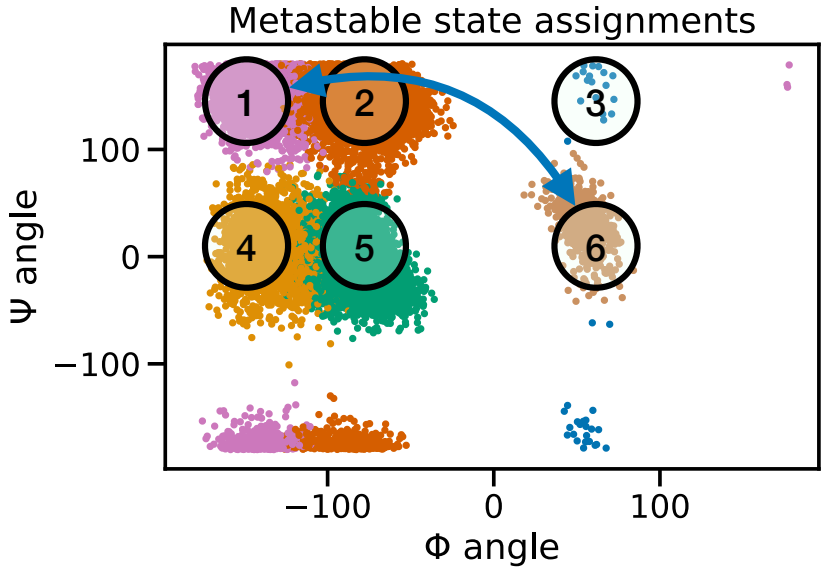


Michael Feig, Journal of Chemical Theory and Computation **2008** 4 (9), 1555-1564, DOI: 10.1021/ct800153n

How to extract dynamics from simulations

1. Simulate system
2. Find good coordinates (a.k.a. collective variables. CVs)
3. Find metastable states (1, 2, …)
4. Estimate rates

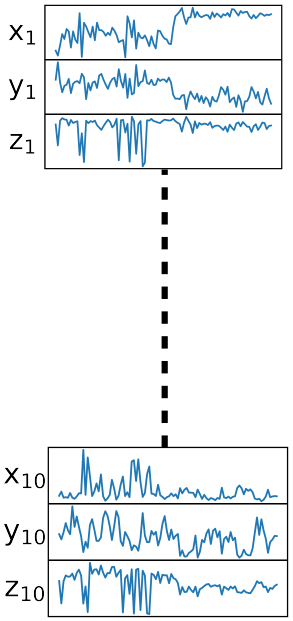




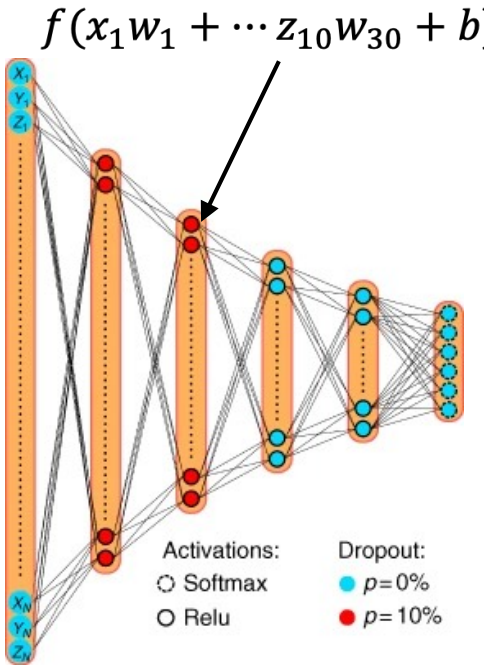
Deep learning for conformational dynamics

Deep learning (DL) can find collective variables

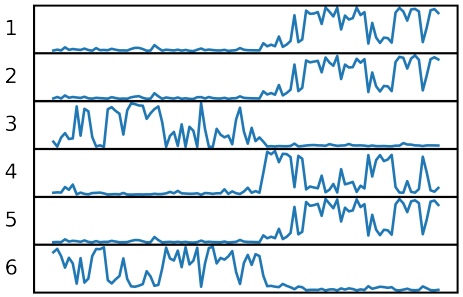
Inputs:
trajectory batch



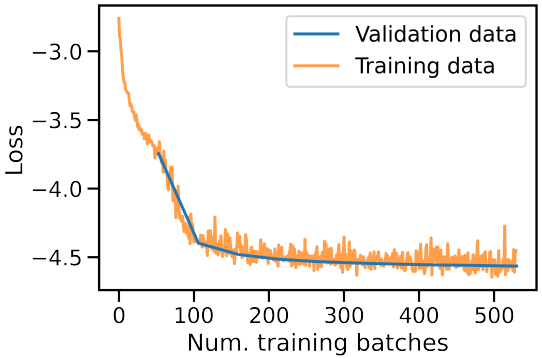
Network:



Outputs:
metastable states

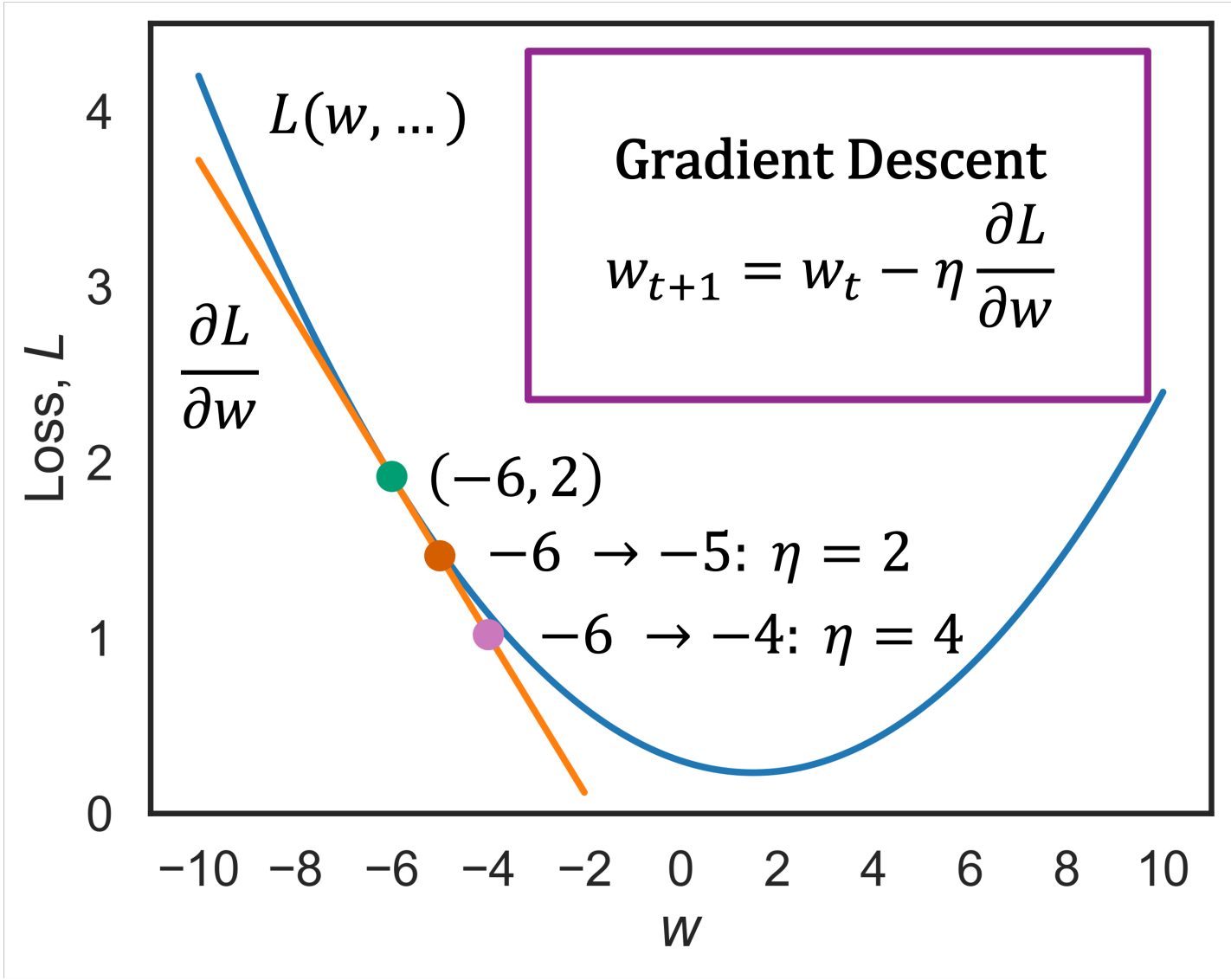


Training:
minimise
'quickness' of rates



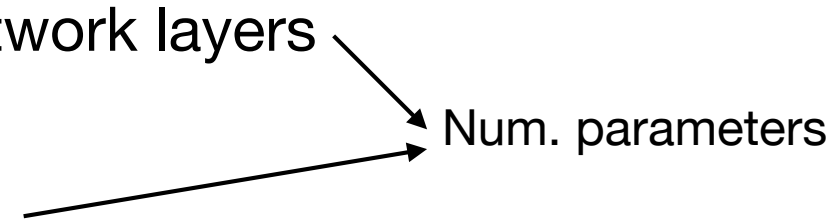
Mardt, A. *et al. Nat Comm* **9**, 5 (2018). https://doi.org/10.1038/s41467-017-02388-1

DL models use gradient descent to adjust weights and biases



Training requires making modelling choices

1. Train many models with different *hyper-parameters*:
    1. numbers of network layers
    2. width of layers
    3. learning rates
    4. other choices e.g., batch sizes, pre-processing steps etc.
2. Pick best set of hyper-parameters
3. Train model several times to get error estimate

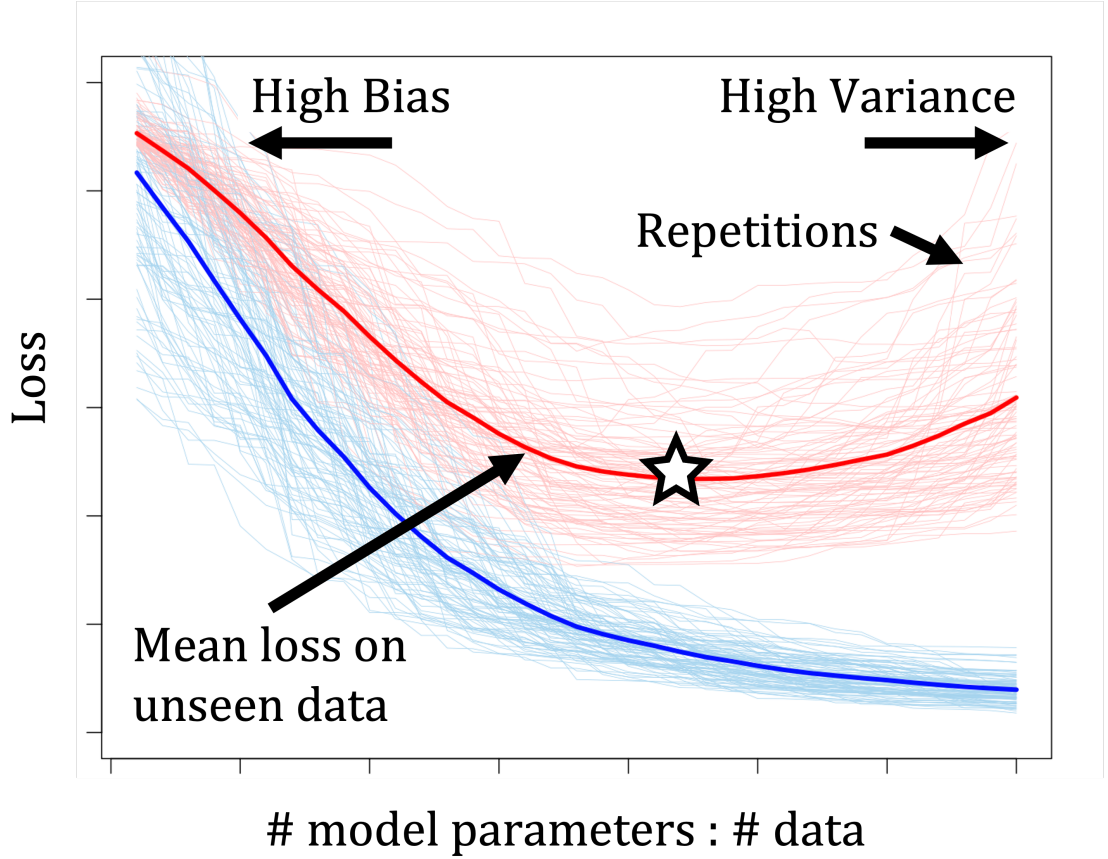


Two problems with training

1. **Gradient descent is slow: 10 - 100s of training epochs**

DALL-E 2: “non-binary scientist running out of time, looking annoyed”
https://labs.openai.com/

2. **Optimal number of parameters? Num. layers? Width of layers?**



Hastie, T., et. al.., 2009. The elements of statistical learning (Vol. 2, pp. 1-758). New York: springer., Chapter 7

Deep online learning

Online algorithms use one pass of the data

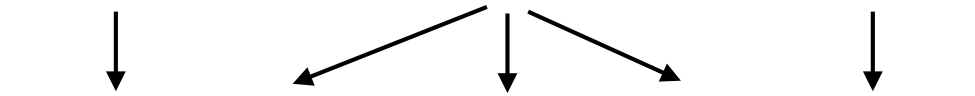
Batch:

$$s^2 = \left(\sum_i (x_i - \mu)^2 \right) \frac{1}{n-1}$$

Online: **Vs**

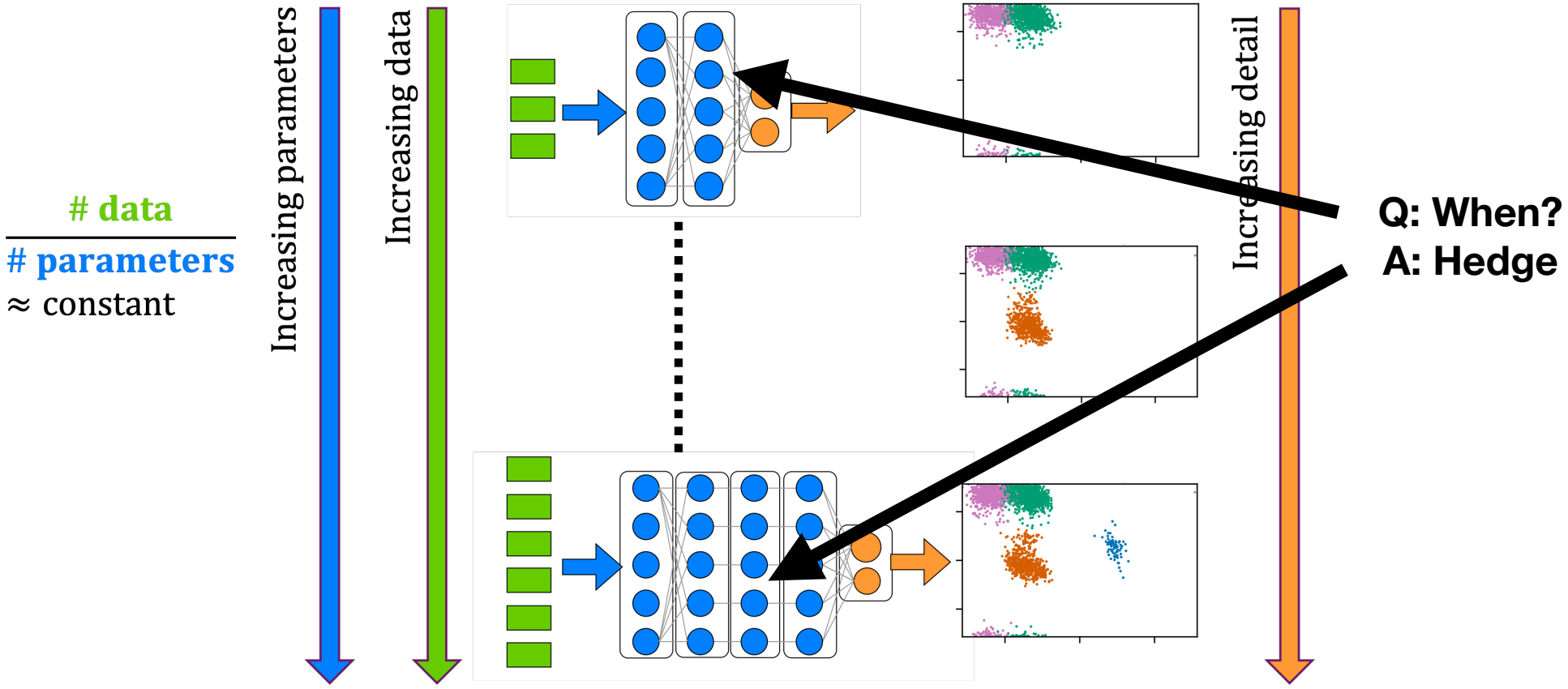
$$s_i^2 = \left(\sum_i x_i^2 - \frac{(\sum_i x_i)^2}{n_i} \right) \frac{1}{n_i - 1}$$

Input: **Calculations:** **Output:**



x	n	$\sum_i x_i$	$\sum_i x_i^2$	s^2
3	1	3	9	-
1	2	4	10	2.0
7	3	11	59	9.33

Online deep learning (ODL) exploits the bias-variance tradeoff



Sahoo, D., Pham, Q., Lu, J. and Hoi, S.C., 2017. Online deep learning: Learning deep neural networks on the fly. *arXiv preprint arXiv:1711.03705*.

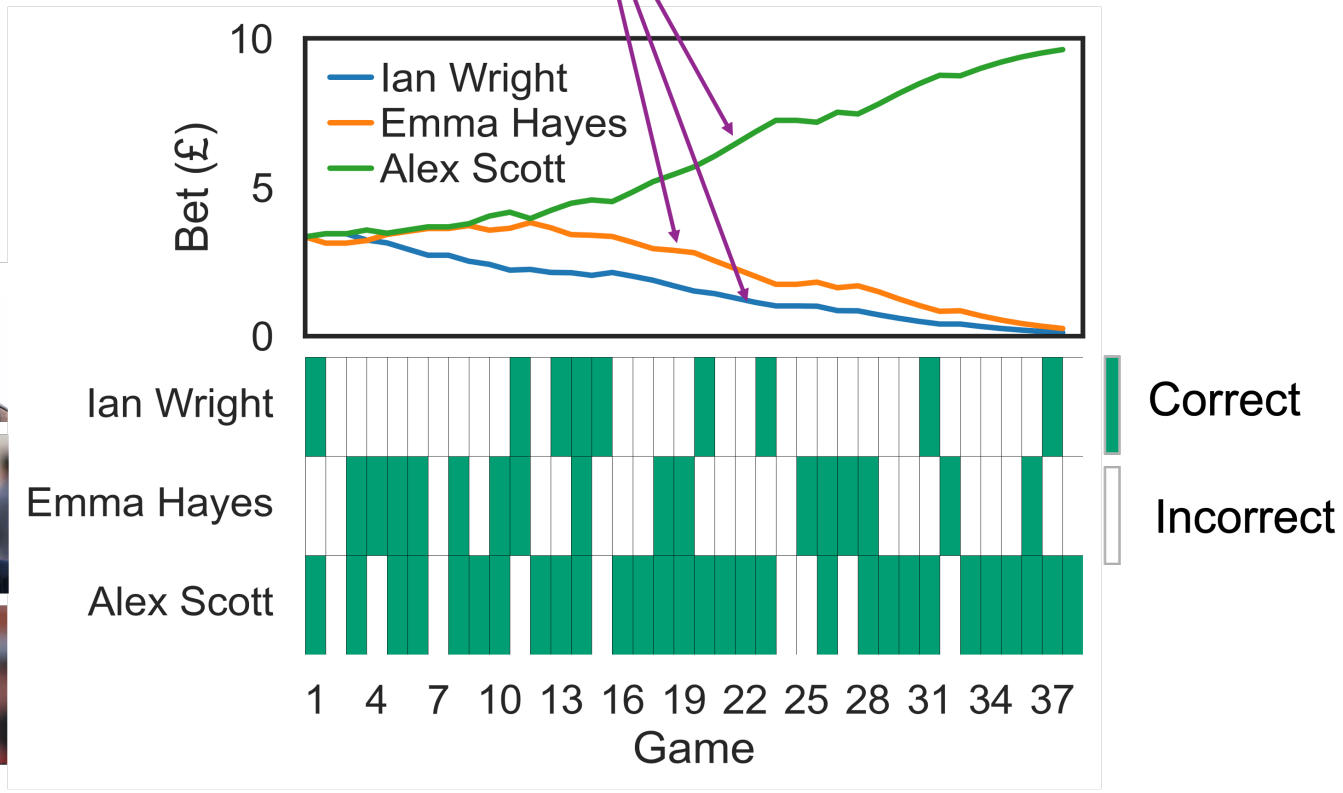
Hedging assigns resources optimally using expert opinion

£10 total bet - how do I use experts optimally?

$$b_{i,\text{new}} = b_{i,\text{old}}\beta^{-l_i}$$

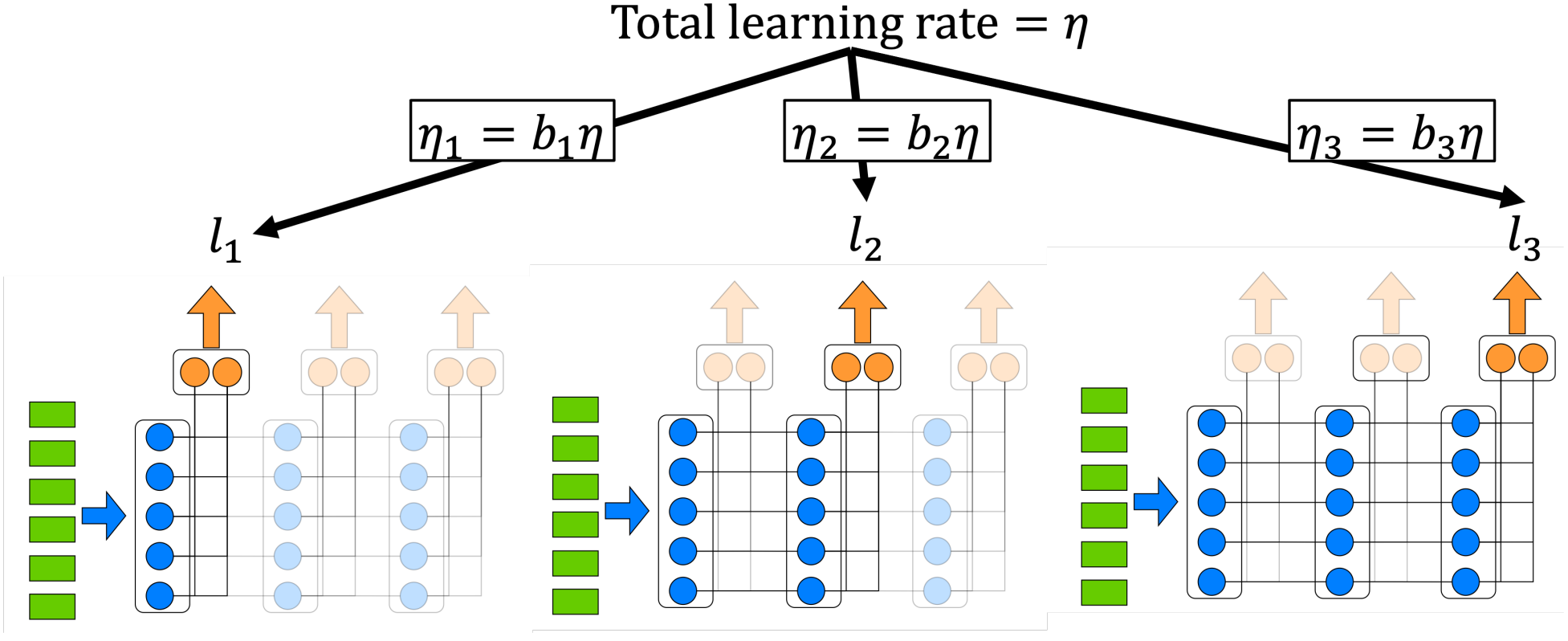
$$b_{i,\text{new}} = b_{i,\text{new}}/\sum b_{i,\text{new}}$$

b_i = proportion bet on pundit i
l_i = amount lost
β = learning rate (= 0.75)



Hedge back propagation apportions learning rates

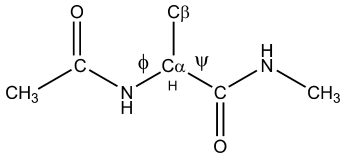
- **1 Large Network = Small + Medium + Large sub-networks**
- **Hedge learning rate across sub-networks,** $b_i \propto \beta^{-l_i}$
- b_1 **large** $\leftrightarrow$ **small data**
- b_2 **large** $\leftrightarrow$ **medium data**
- b_3 **large** $\leftrightarrow$ **large data**



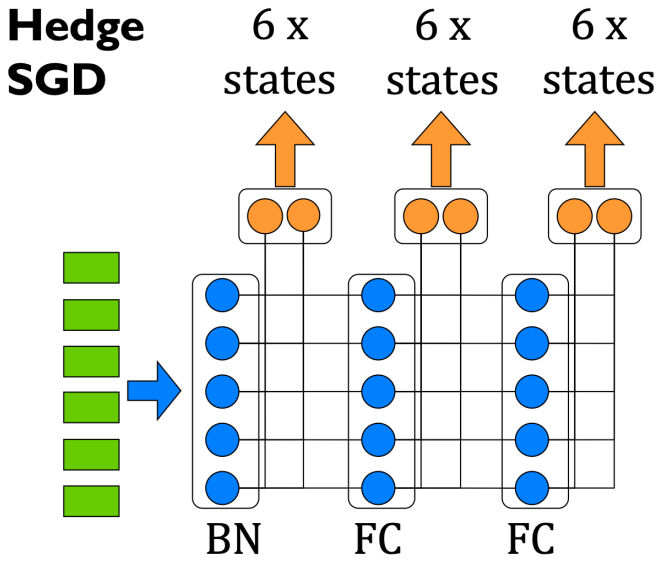
Results

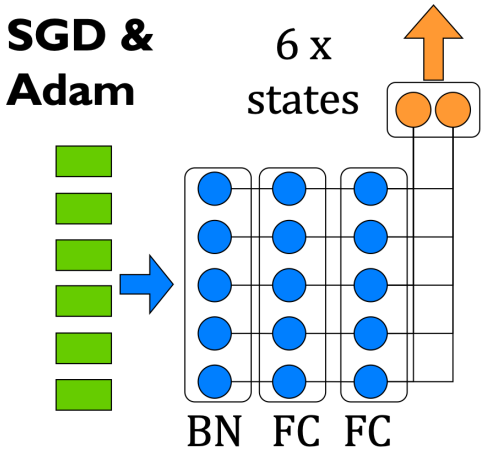
Experimental setup

- **Data:** $750\mu s$ **heavy atom coordinates alanine dipeptide**



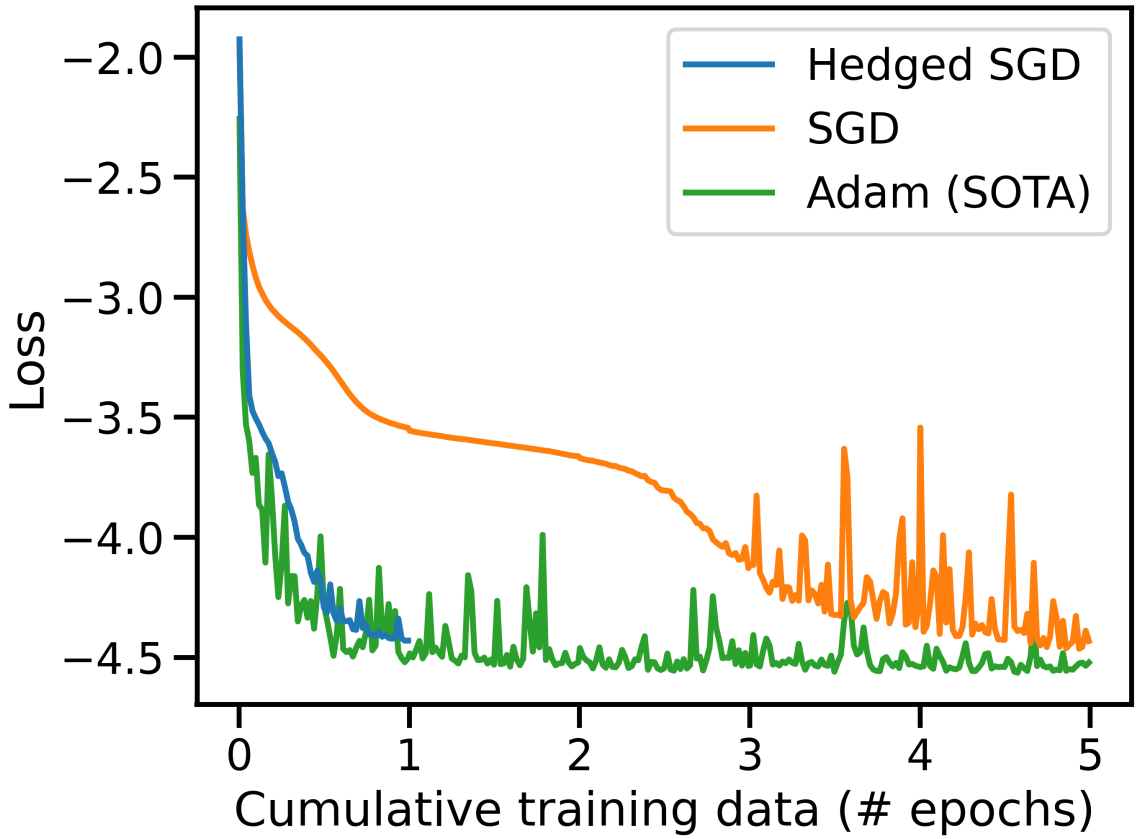
- **Training/Validation:** $525\mu s : 225\mu s$
- **Learning rates:**
  - $\eta = 1.5 \times 10^{-2}$ (weights update)
  - $\beta = 0.98$ (expert update)
- **1 batch normalisation (BN) + 2 x fully-connected (FC)**
- **Layer width:** 100
- **Output:** 6 metastable states
- **Training:**
  - Hedge SGD (new)
  - SGD (existing)
  - Adam (existing, SOTA)



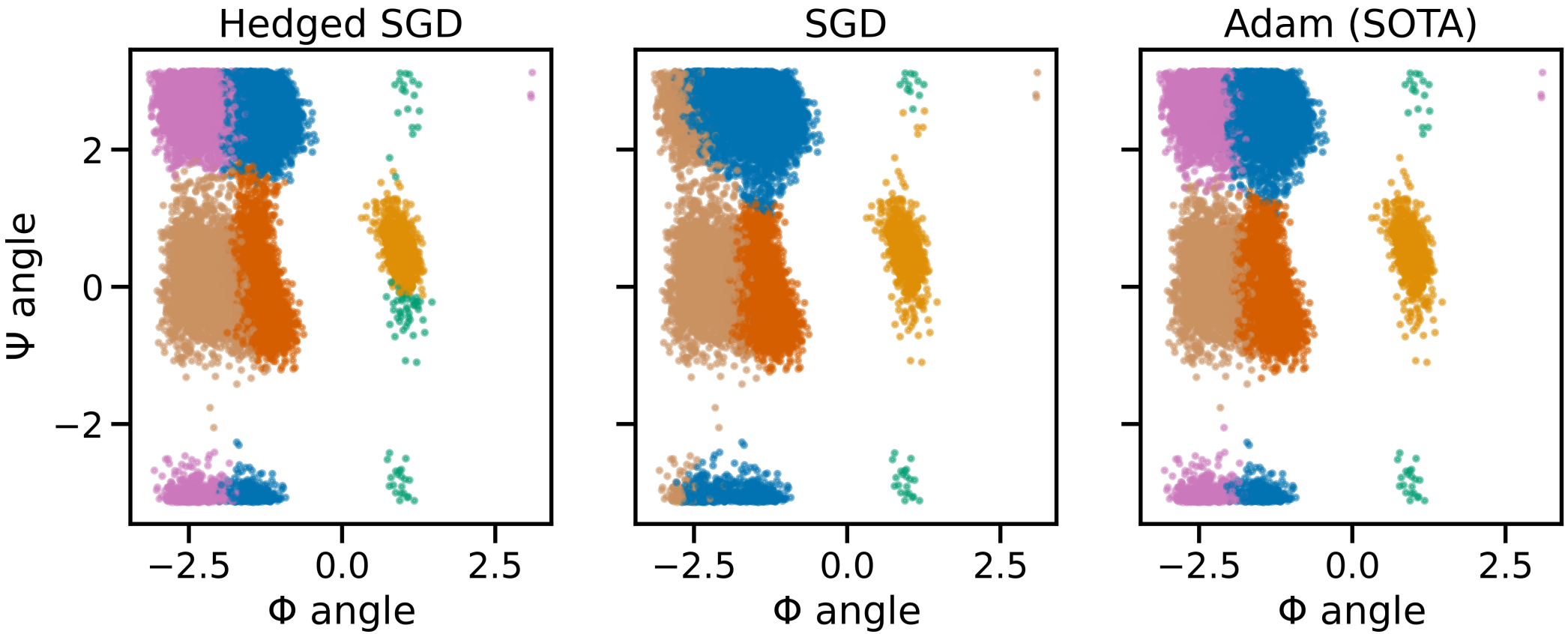


Results: validation loss

- Hedged SGD >SGD
- Hedged SGD $\simeq$ SOTA (Adam)
- Adam adapts η with data

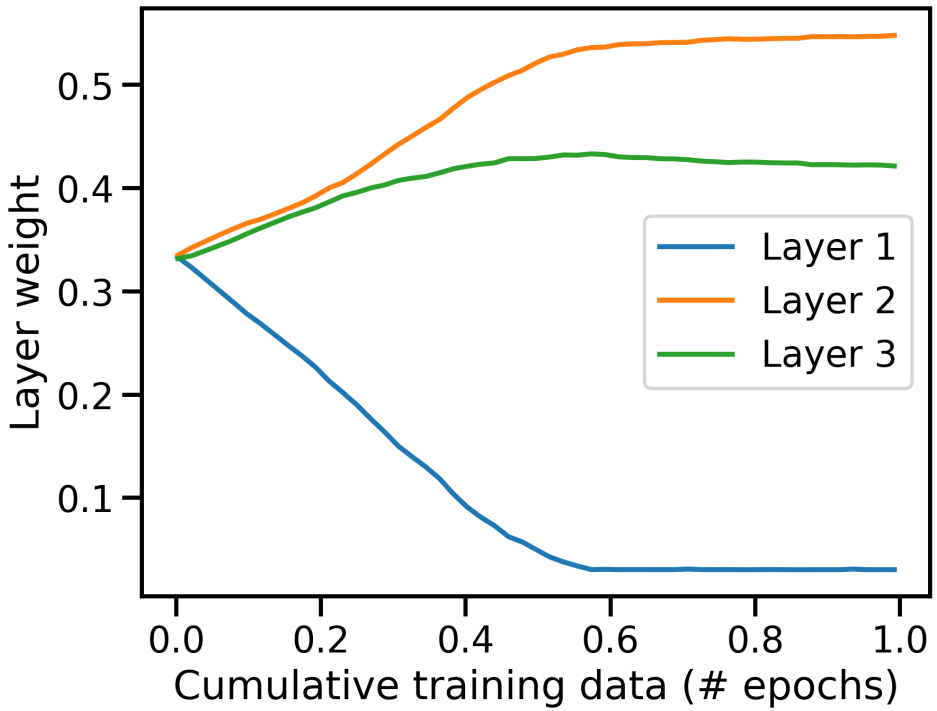


Results: Metastable assignments

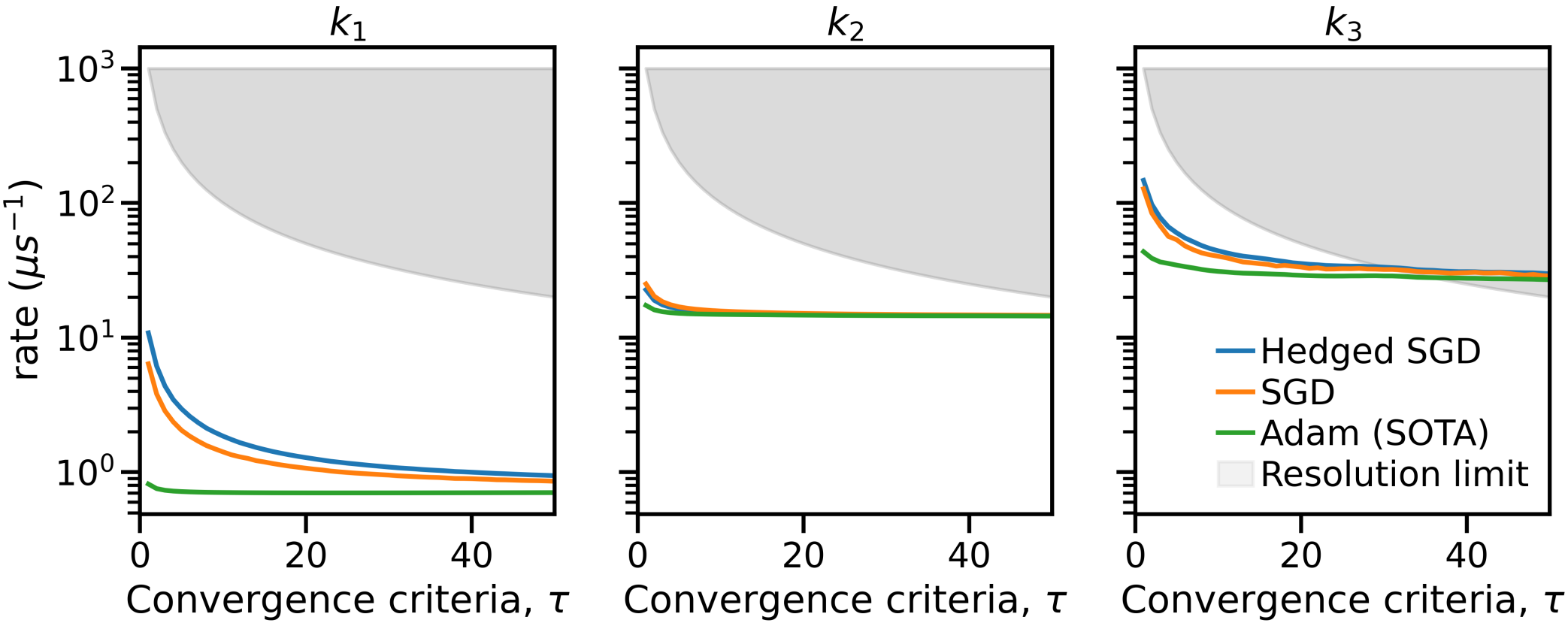


Results: weights

- Weights tell how predictive each layer is.
- 2nd layer most predictive.
- *Can* use only 2 layers and still be ok.



Results: rates



Conclusions and outlook

Conclusions

- Hedged stochastic gradient descent (SGD):
  - performs at least as well as SGD for rates and metastable states
  - trains as fast as Adam (SOTA)
  - provides insight into how many layers are important
- Problems:
  - Numerically unstable
  - Fails with large numbers of layers

Outlook

- Need better test case - alanine dipeptide is **too easy**
  - Protein folding
  - Pocket dynamics
- Hedging can be applied to:
  - Determine # of metastable states
  - Update pre-trained model (transfer learning)
  - Time sensitive pipelines (adaptive sampling)

Thanks

- Toni Mey, supervision
- Redesign Science, funding and supervision
- Ben Leimkuhler and Tiffany Vlaar for discussion