Machine Learning Methods for Neural Data Analysis

Lecture 15: Beyond SLDS, Stochastic RNNs

Scott Linderman

STATS 220/320 (NBIO220, CS339N). Winter 2021.

Announcements

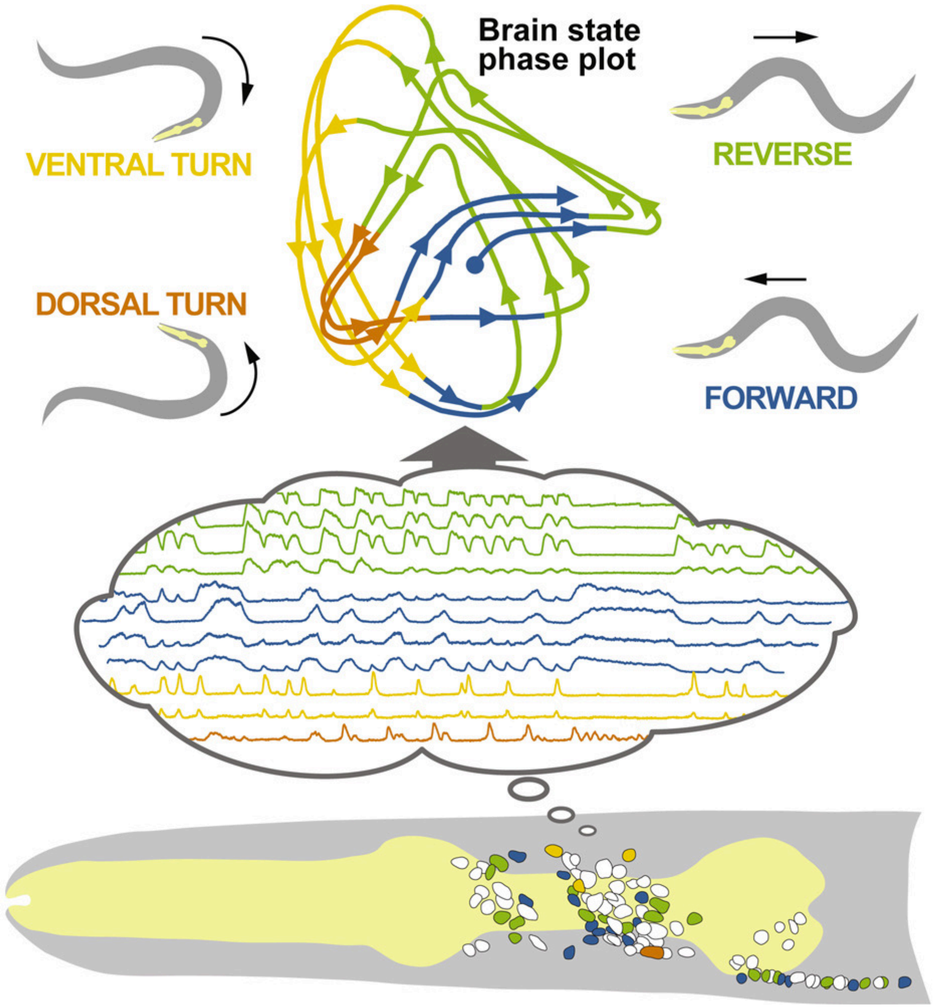
- Lab 8 test code
- Final projects

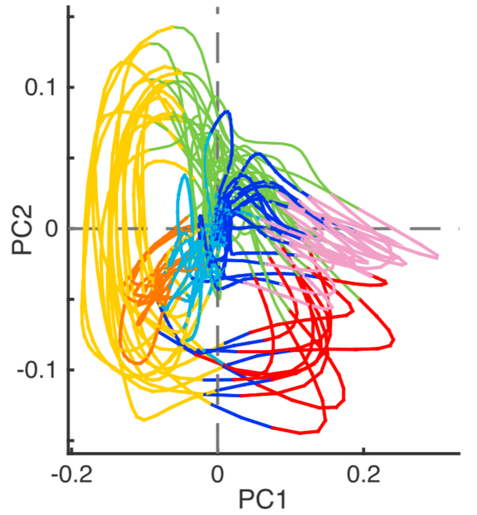
Agenda

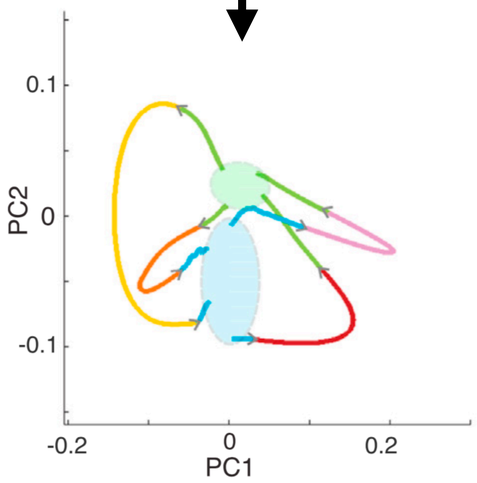
- Generalizing the SLDS
- Stochastic RNNs

Generalizing the SLDS

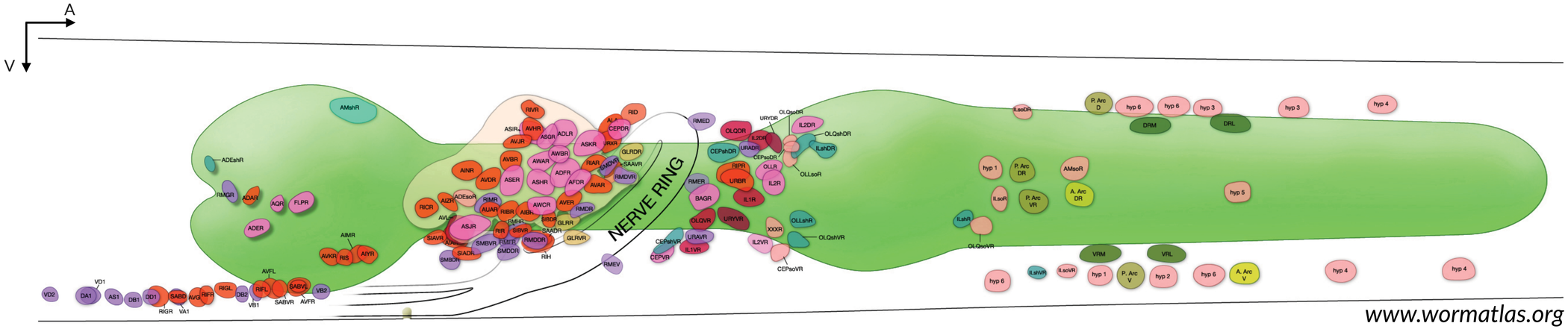
Previous work suggests that this neural activity lies on a low dimensional manifold partitioned by behavior

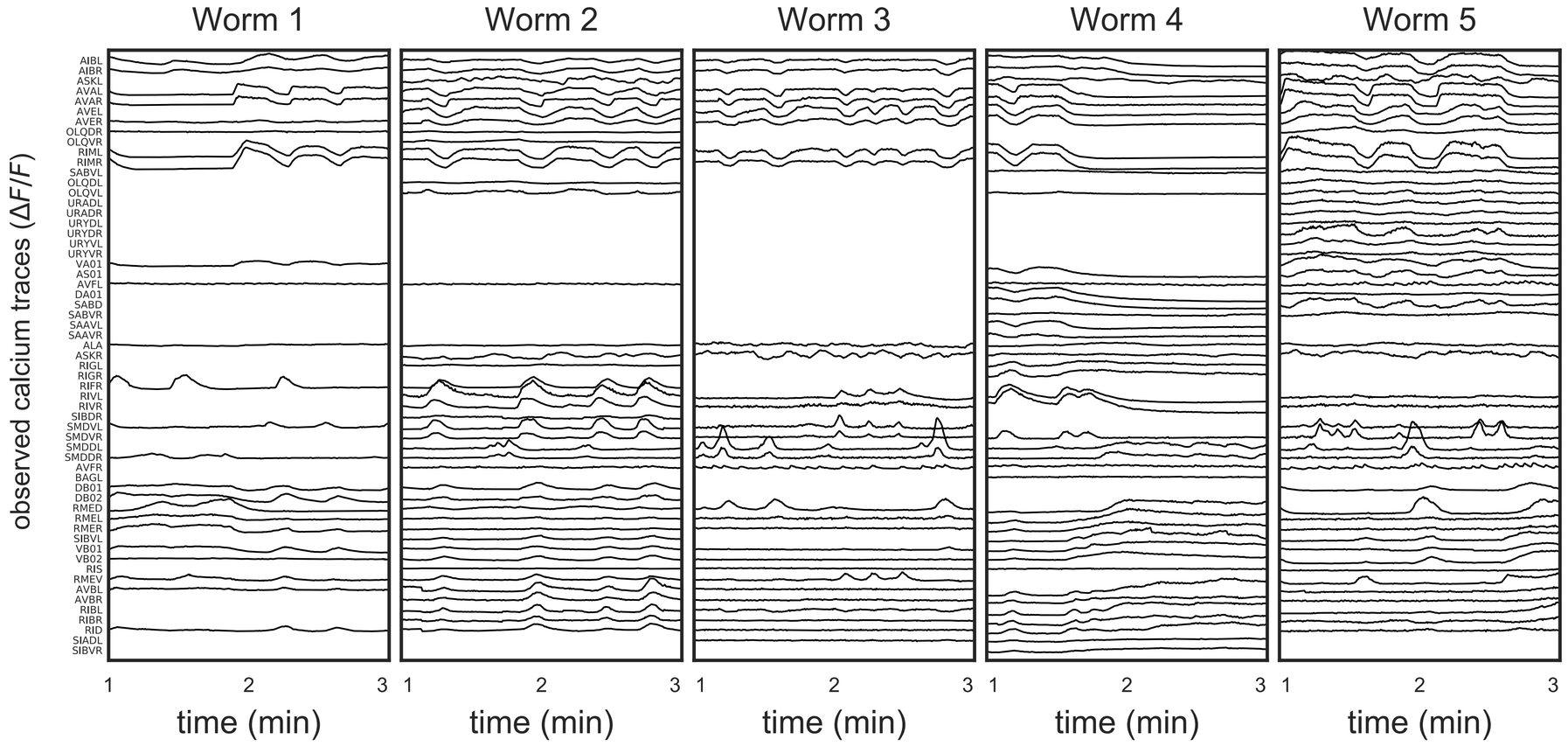






Kato et al (2015)


A
V
NERVE RING
www.wormatlas.org


Worm 1
Worm 2
Worm 3
Worm 4
Worm 5
observed calcium traces (ΔF/F)
AIBL
AIBR
ASKL
AVAL
AVAR
AVEL
AVER
OLQDR
OLQVR
RIML
RIMR
SABVL
OLQDL
OLQVL
URADL
URADR
URYDL
URYDR
URYVL
URYVR
VA01
AS01
AVFL
DA01
SABD
SABVR
SAAVL
SAAVR
ALA
ASKR
RIGL
RIGR
RIFR
RIVL
RIVR
SIBDR
SMDVL
SMDVR
SMDDL
SMDDR
AVFR
BAGL
DB01
DB02
RMED
RMEL
RMER
SIBVL
VB01
VB02
RIS
RMEV
AVBL
AVBR
RIBL
RIBR
RID
SIADL
SIBVR
1
2
3
time (min)

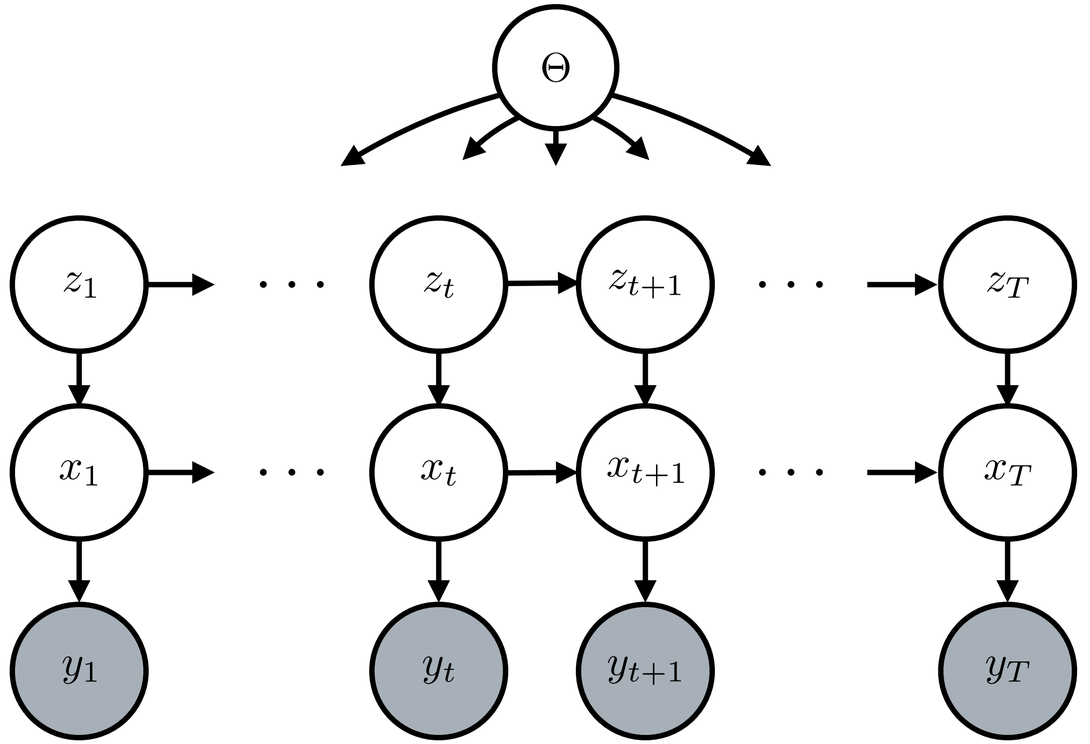
Switching Linear Dynamical Systems

Global Parameters

Discrete Latent States

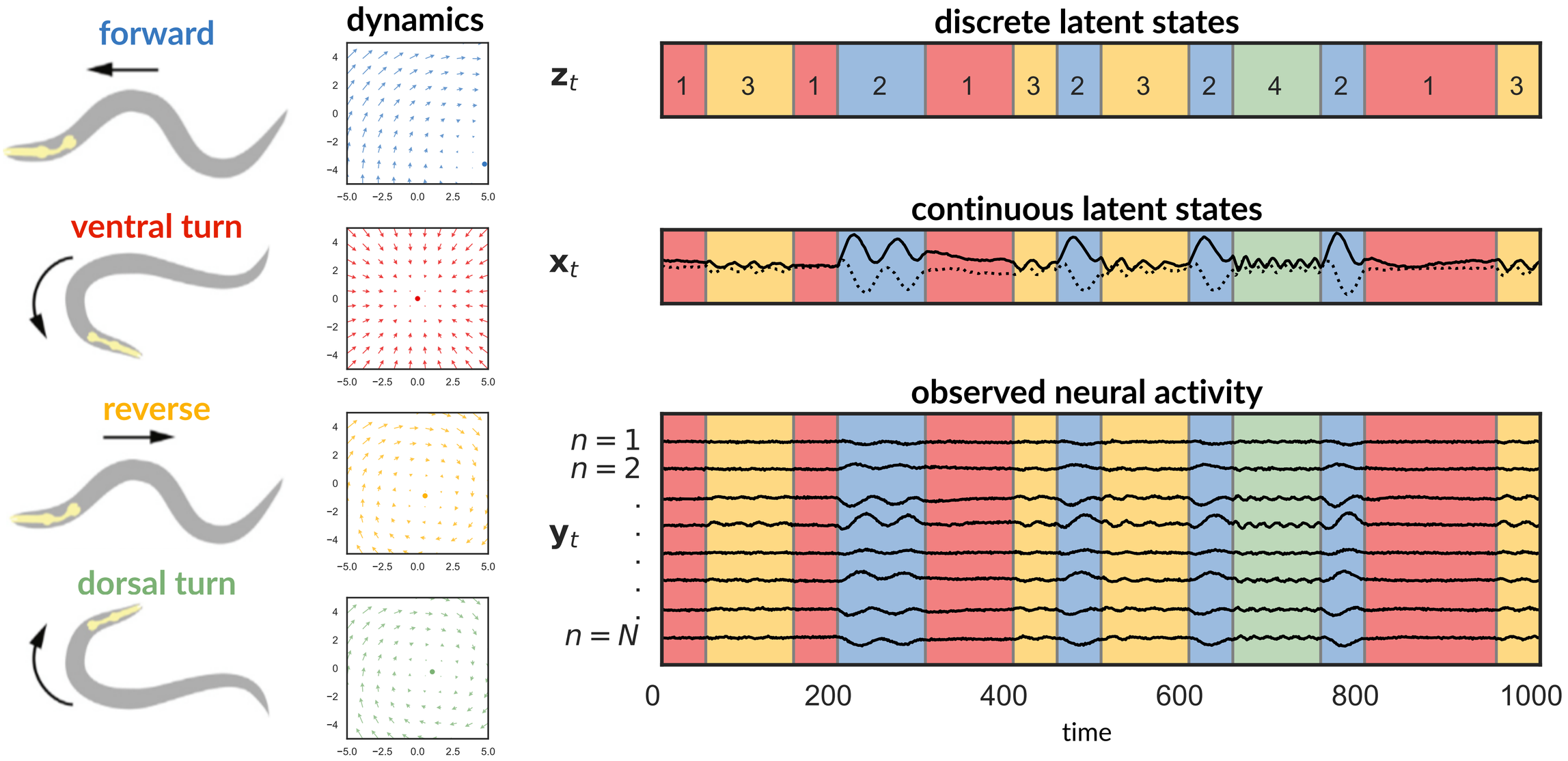
Continuous Latent States

Observed Neural Activity ($\Delta F/F_0$)

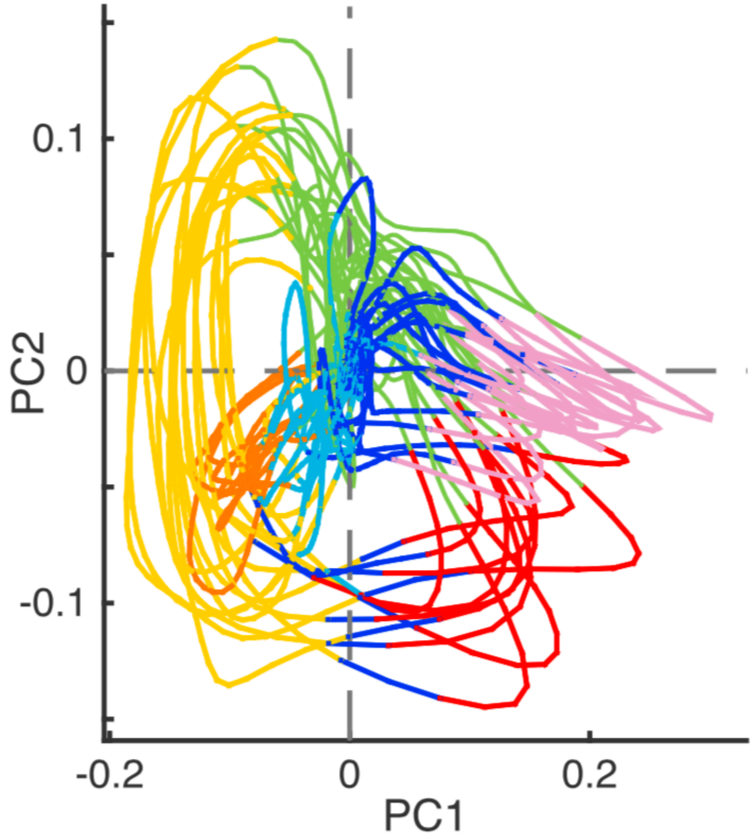


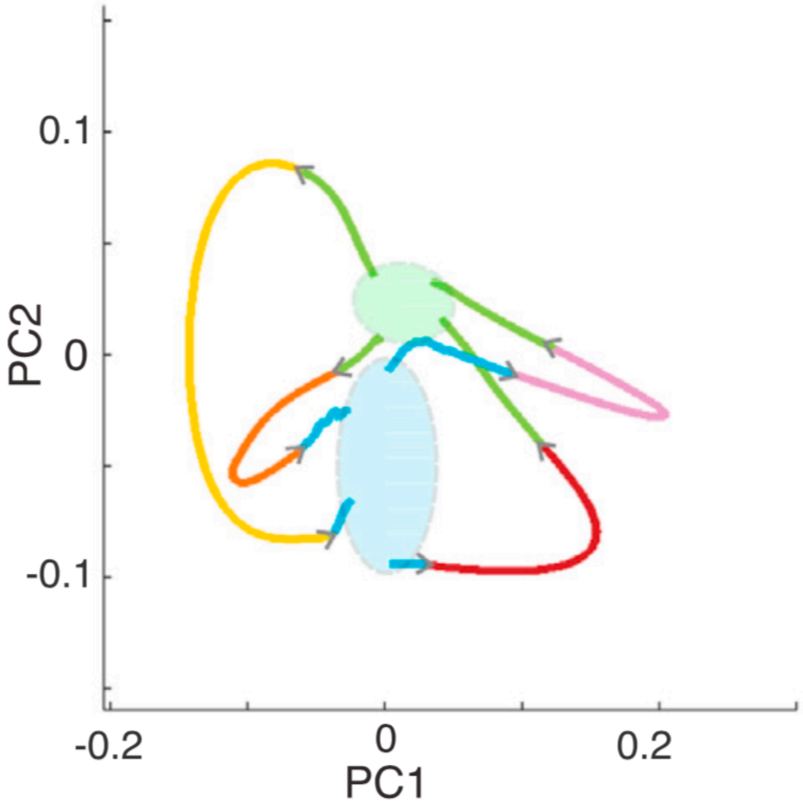
= latent = observed = dependency

Building a probabilistic model of neural data



We expect discrete states to be used in localized regions of the low-dimensional manifold





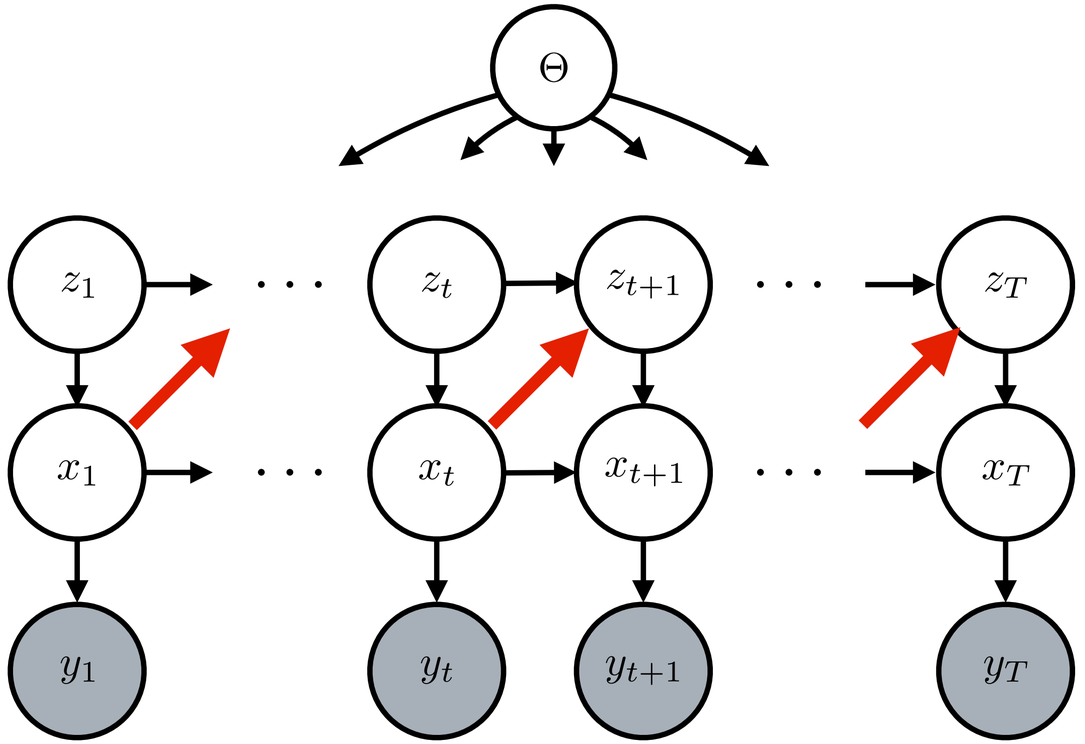
“Recurrent” Switching Linear Dynamical System (rSLDS)

Global Parameters

Discrete Latent States

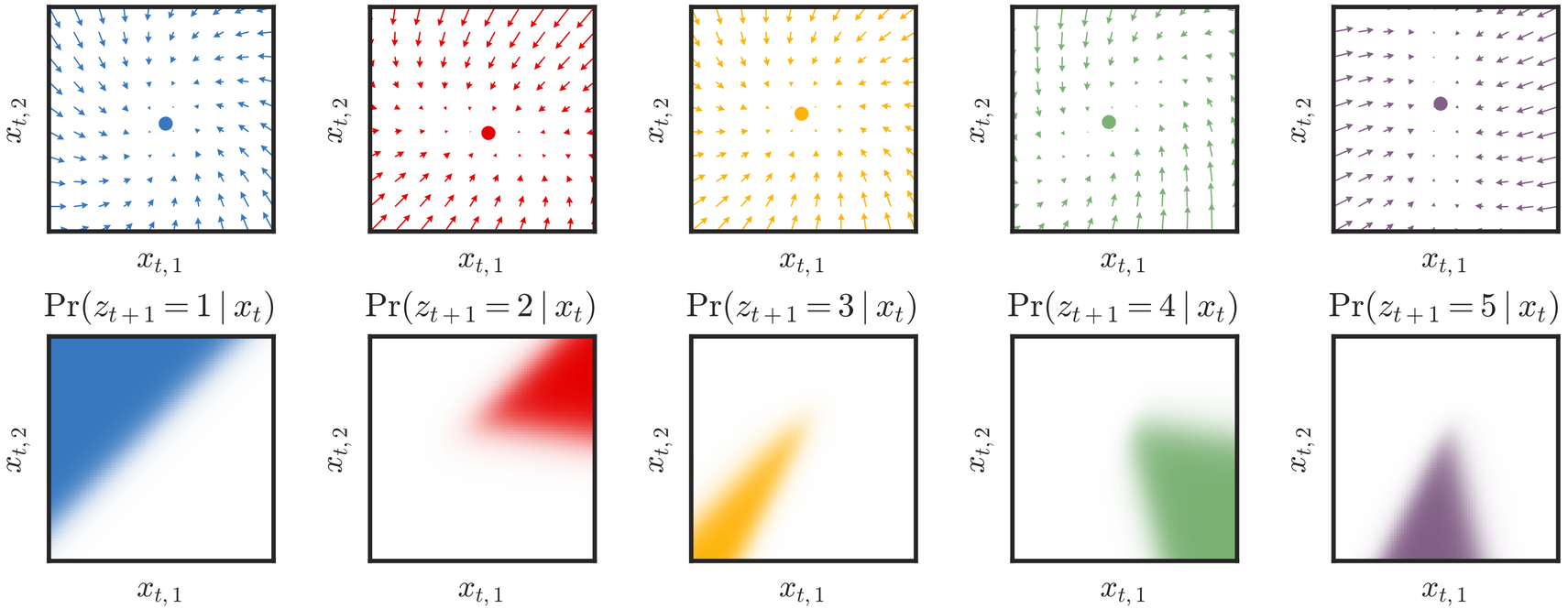
Continuous Latent States

Observed Neural Activity (Differenced $\Delta F/F_0$)



Barber (2006)
Linderman et al. (2017)

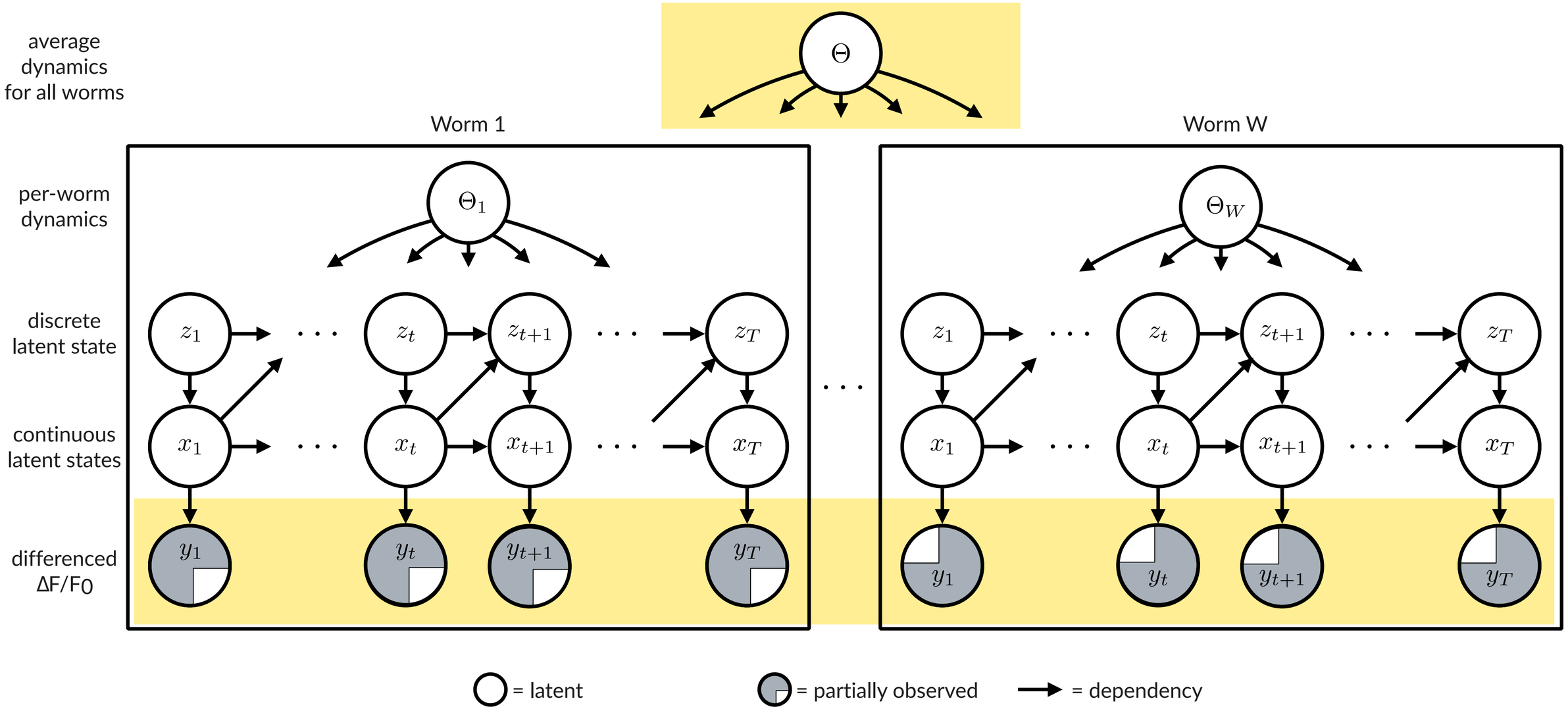
Recurrent dependencies carve up continuous space



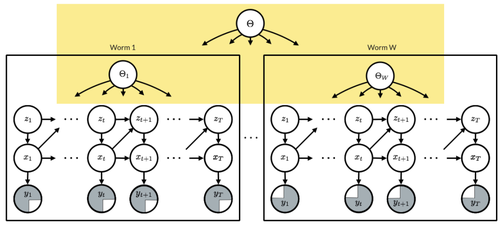
See also:

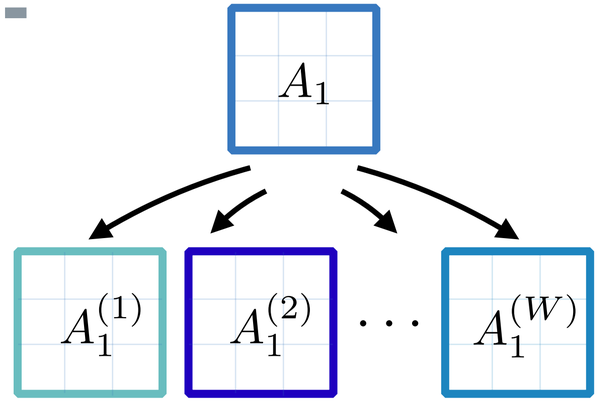
- *Piecewise affine models*
- *Hybrid systems*
- *Tree-structured rSLDS (Nassar et al, 2019)*

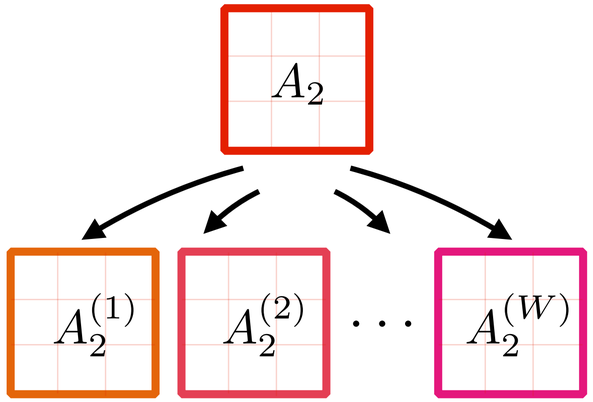
Extend hierarchically to capture individual variability and learn from many partial recordings



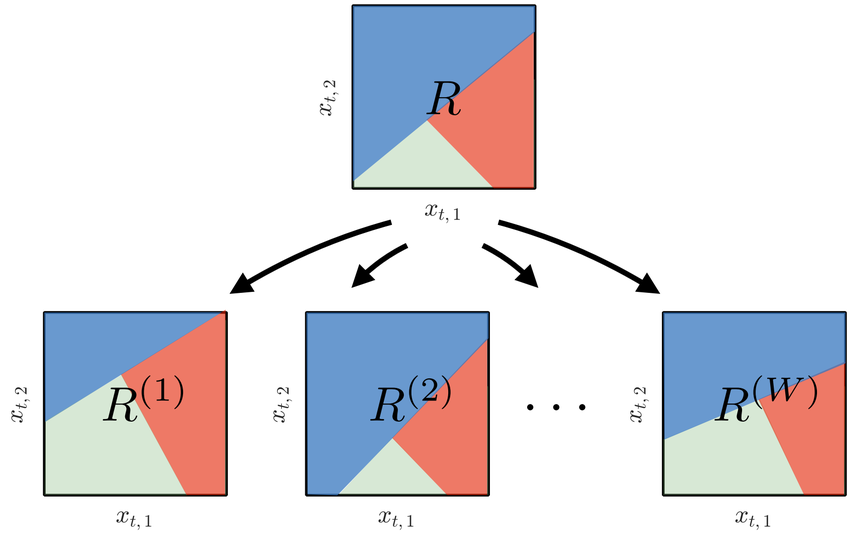
Implementing the hierarchical prior



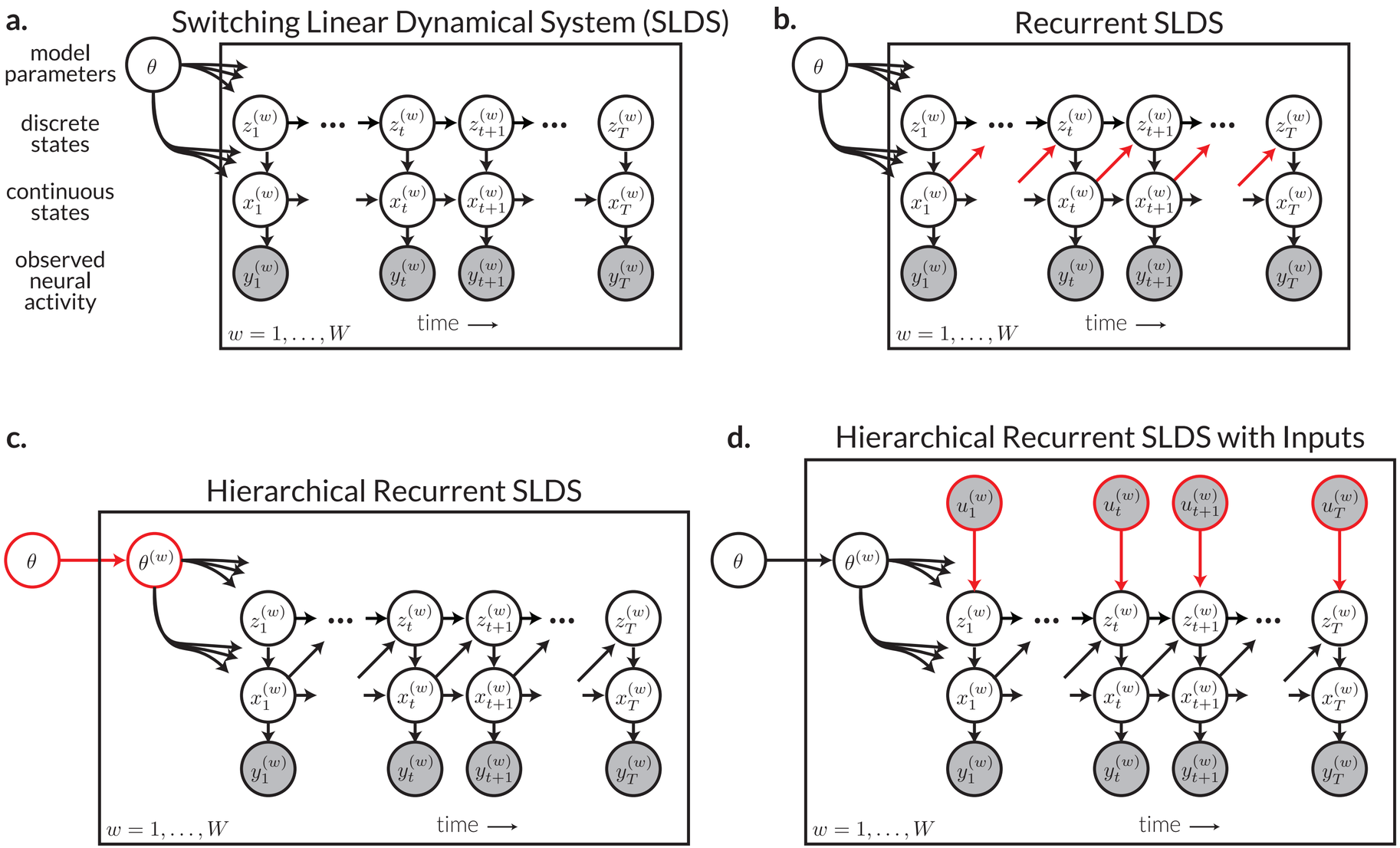




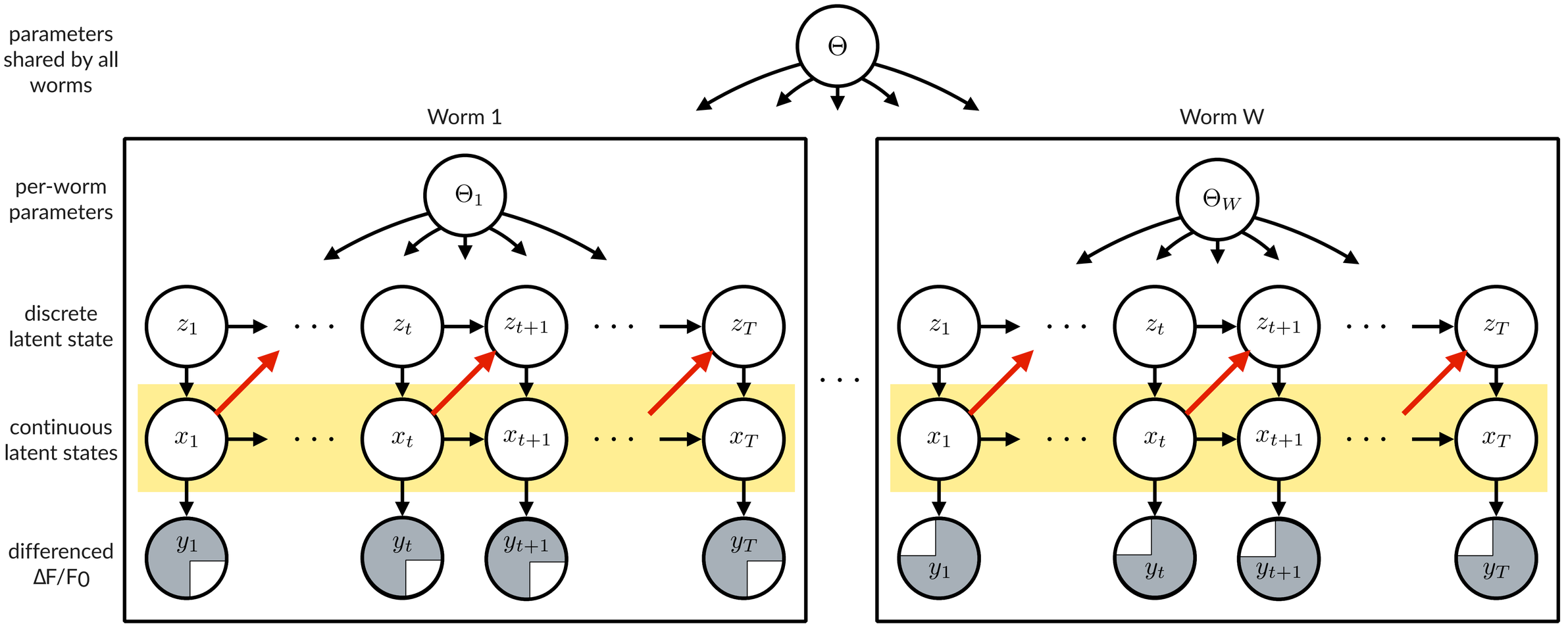
$$\mathrm{vec}(A_k^{(w)}) \sim \mathcal{N}(\mathrm{vec}(A_k), \sigma^2 I)$$



$$R_k^{(w)} \sim \mathcal{N}(R_k, \eta^2 I)$$

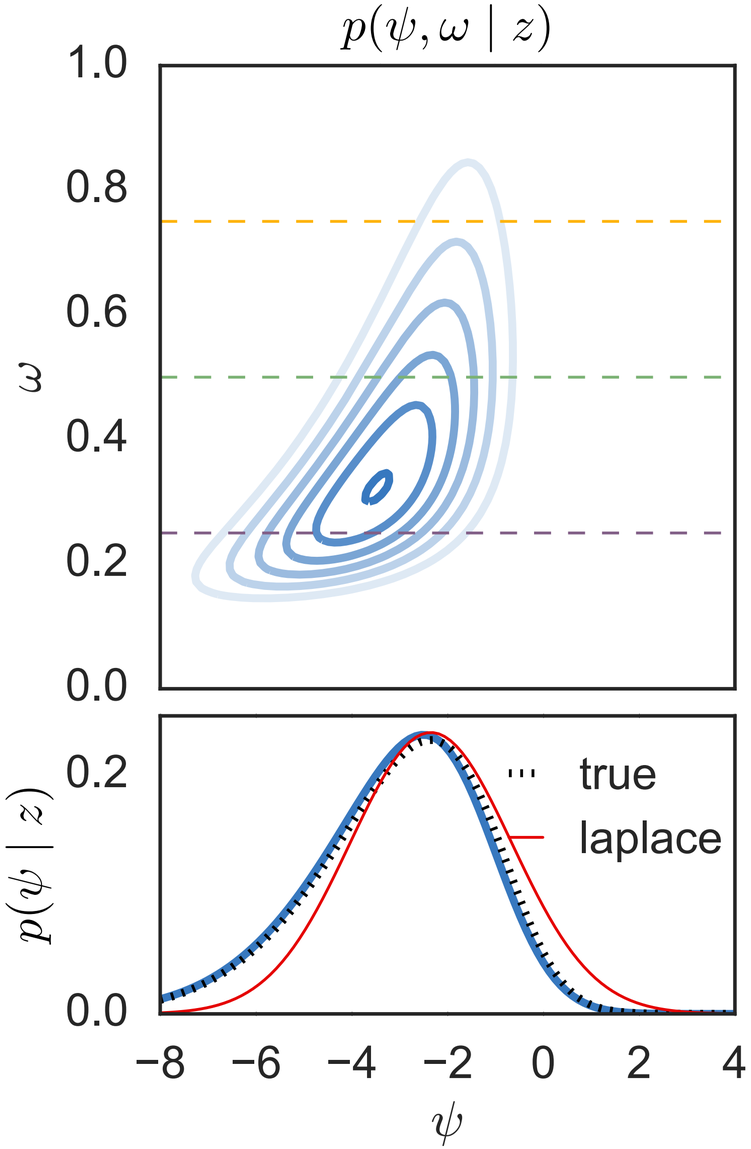

a.
Switching Linear Dynamical System (SLDS)
model parameters
discrete states
continuous states
observed neural activity
θ
$z_1^{(w)}$ ... $z_t^{(w)}$ $z_{t+1}^{(w)}$ ... $z_T^{(w)}$
$x_1^{(w)}$ $x_t^{(w)}$ $x_{t+1}^{(w)}$ $x_T^{(w)}$
$y_1^{(w)}$ $y_t^{(w)}$ $y_{t+1}^{(w)}$ $y_T^{(w)}$
$w = 1, \ldots, W$
time ⟶
b.
Recurrent SLDS
θ
$z_1^{(w)}$ ... $z_t^{(w)}$ $z_{t+1}^{(w)}$ ... $z_T^{(w)}$
$x_1^{(w)}$ $x_t^{(w)}$ $x_{t+1}^{(w)}$ $x_T^{(w)}$
$y_1^{(w)}$ $y_t^{(w)}$ $y_{t+1}^{(w)}$ $y_T^{(w)}$
$w = 1, \ldots, W$
time ⟶
c.
Hierarchical Recurrent SLDS
θ
$\theta^{(w)}$
$z_1^{(w)}$ ... $z_t^{(w)}$ $z_{t+1}^{(w)}$ ... $z_T^{(w)}$
$x_1^{(w)}$ $x_t^{(w)}$ $x_{t+1}^{(w)}$ $x_T^{(w)}$
$y_1^{(w)}$ $y_t^{(w)}$ $y_{t+1}^{(w)}$ $y_T^{(w)}$
$w = 1, \ldots, W$
time ⟶
d.
Hierarchical Recurrent SLDS with Inputs
θ
$\theta^{(w)}$
$u_1^{(w)}$ $u_t^{(w)}$ $u_{t+1}^{(w)}$ $u_T^{(w)}$
$z_1^{(w)}$ ... $z_t^{(w)}$ $z_{t+1}^{(w)}$ ... $z_T^{(w)}$
$x_1^{(w)}$ $x_t^{(w)}$ $x_{t+1}^{(w)}$ $x_T^{(w)}$
$y_1^{(w)}$ $y_t^{(w)}$ $y_{t+1}^{(w)}$ $y_T^{(w)}$
$w = 1, \ldots, W$
time ⟶

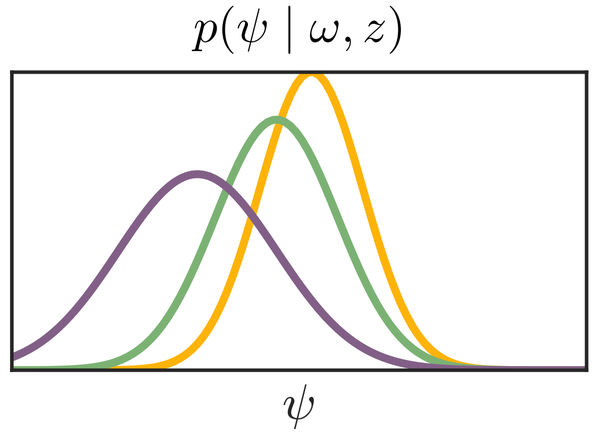
Recurrent dependencies pose challenges



Variational updates are no longer closed form

Pólya-gamma augmentation for Bayesian logistic regression





$$\left.\begin{aligned} \psi &\sim \mathcal{N}(J, h) \\ z \mid \psi &\sim \mathrm{Bern}(\sigma(\psi)) \\ \omega \mid \psi &\sim \mathrm{PG}(1, \psi) \end{aligned}\right\} \quad \psi \mid z, \omega \sim \mathcal{N}\left(J + \omega, h + (z - \tfrac{1}{2})\right)$$

Polson, Scott, and Windle, 2013

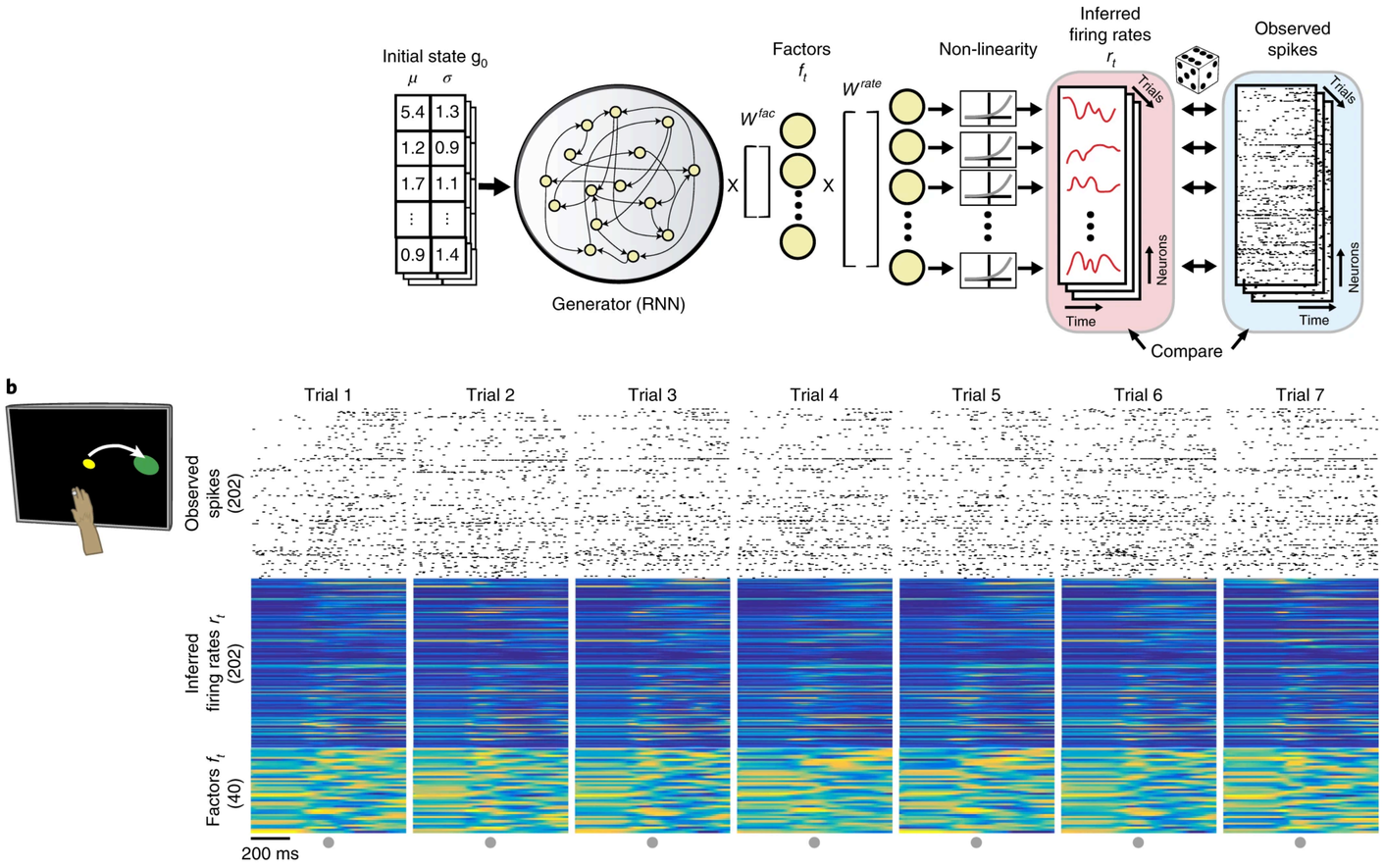
https://github.com/slinderman/pypolyagamma

Stochastic RNNs

Stochastic RNNs

LFADS: Latent Factor Analysis for Dynamical Systems

- LFADS uses a recurrent neural network (the **generator**) to model nonlinear dynamics of neural activity.
- In the basic model, the RNN has **deterministic dynamics** with a **random initial condition.**
- The RNN state is mapped through a **GLM** to obtain firing rates for a **Poisson model.**

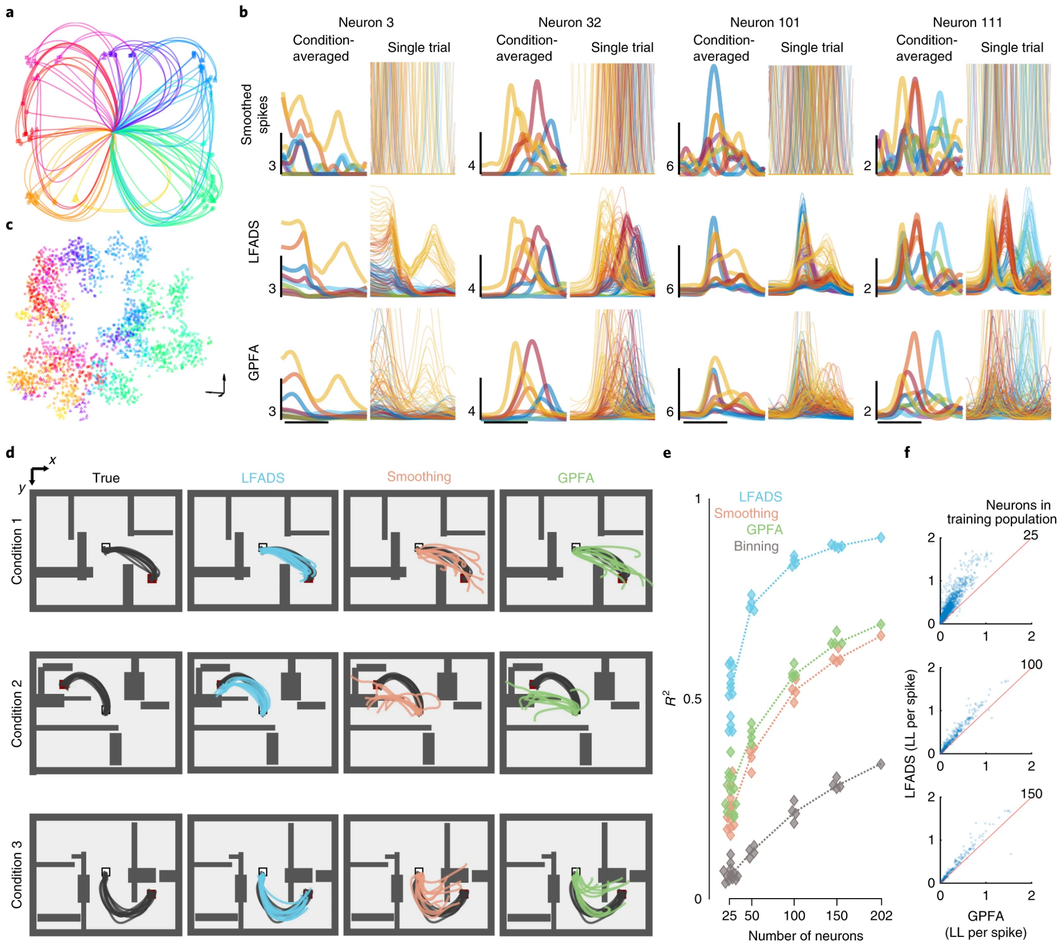


Pandarinath et al (2018)

Stochastic RNNs

LFADS: Latent Factor Analysis for Dynamical Systems

- LFADS learns accurate **single-trial firing rates** and achieves **state-of-the-art decoding performance** on monkey reaching tasks (Recall Lab 6).

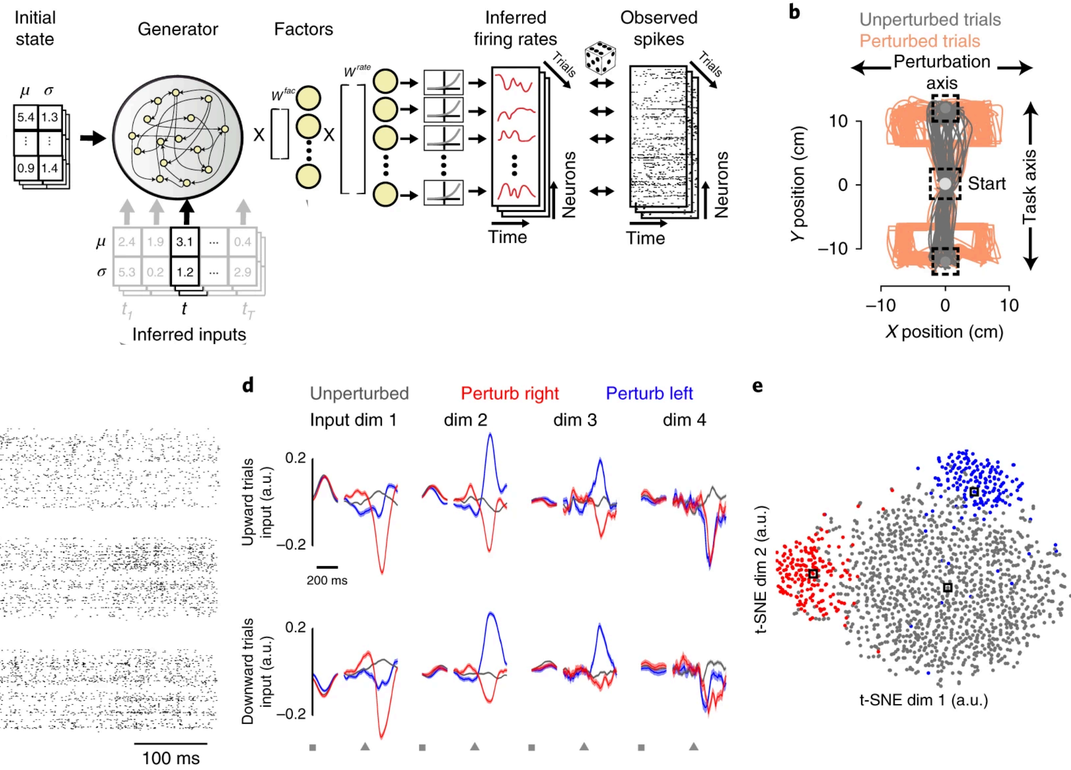


Pandarinath et al (2018)

Stochastic RNNs

LFADS: Latent Factor Analysis for Dynamical Systems

- LFADS can be augmented with **inferred inputs** to uncovers the presence, identity, and timing of **unexpected changes** in the dynamics.
- For example, in trials where the **cursor was randomly perturbed** to the right or left, inputs capture corresponding changes in neural activity.



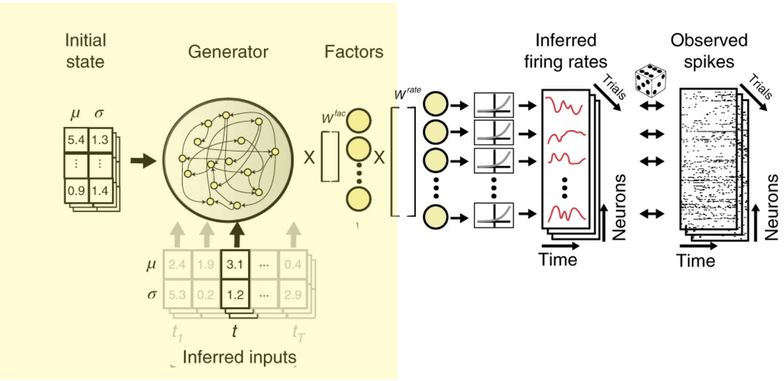
Pandarinath et al (2018)

Stochastic RNNs

The LFADS probabilistic model

- Let:
  - $x_t \in \mathbb{R}^D$ denote the latent state at time t
  - $x_0 \in \mathbb{R}^D$ denote the random initial condition
  - $u_t \in \mathbb{R}^U$ denote the latent input at time t, and
  - Θ denote the parameters.
- The latent states are **deterministic functions** of the initial condition and the inputs,

$$
\begin{aligned}
x_t &= f(x_{t-1}, u_t, \Theta) \\
&= f(f(x_{t-2}, u_{t-1}, \Theta), u_t, \Theta) \\
&= f(\cdots f(f(x_0, u_1, \Theta), u_2, \Theta) \cdots) \\
&\triangleq f_t(x_0, u_{1:t}, \Theta)
\end{aligned}
$$



Stochastic RNNs

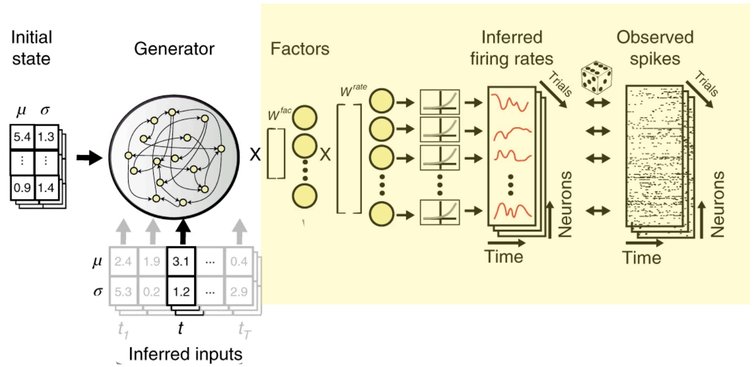
The LFADS probabilistic model

- The output is modeled as a GLM on top of the latent state,

$$\mathbb{E}[y_t] = g(Cx_t + d)$$

- E.g. for spike counts, we commonly use,

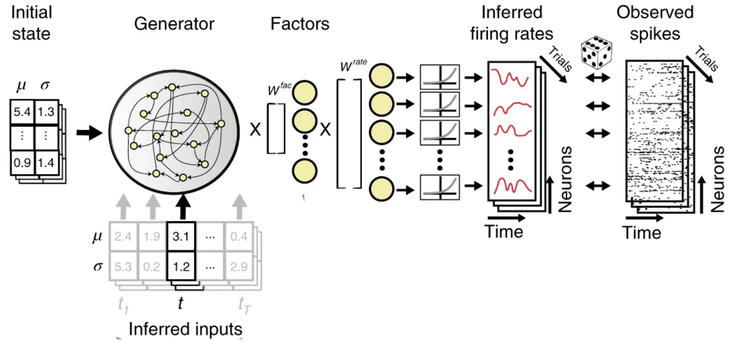
$$y_t \sim \mathrm{Po}\Big(g(Cx_t + d)\Big)$$



Stochastic RNNs

The LFADS probabilistic model

- Assume the initial condition and inputs have standard normal priors.
- The joint distribution is,



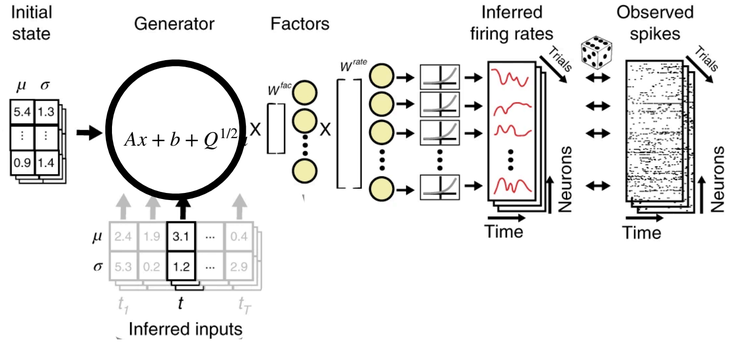
$$p(x_0, u_{1:T}, y_{1:T} \mid \Theta) = \mathcal{N}(x_0 \mid 0, I) \prod_{t=1}^{T} \mathcal{N}(u_t \mid 0, I)\, \mathrm{Po}(y_t \mid g(Cx_t + d))$$

$$= \mathcal{N}(x_0 \mid 0, I) \prod_{t=1}^{T} \mathcal{N}(u_t \mid 0, I)\, \mathrm{Po}\big(y_t \mid g(Cf_t(x_0, u_{1:t}, \Theta) + d)\big)$$

Stochastic RNNs

Poisson LDS as a special case of LFADS

- We can view the **Poisson LDS** (c.f. Macke et al, 2011) as a special case of LFADS with a **linear generator**.



$$x_t \sim \mathcal{N}(Ax_{t-1} + b, Q)$$

$$y_t \sim \mathrm{Po}\Big(g(Cx_t + d)\Big)$$

$$\Longleftrightarrow$$

$$x_t = f(x_{t-1}, u_t, \Theta)$$

$$f(x_{t-1}, u_t, \Theta) = Ax_{t-1} + b + Q^{1/2}u_t$$

$$u_t \sim \mathcal{N}(0, I)$$

$$y_t \sim \mathrm{Po}\Big(g(Cx_t + d)\Big)$$

Stochastic RNNs

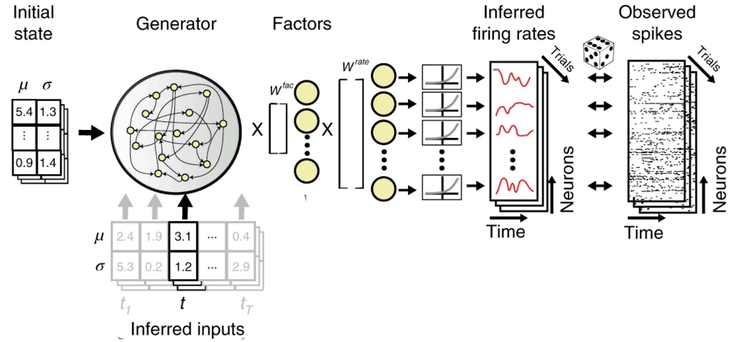
LFADS learning and inference

- How to learn the parameters Θ and infer the latent variables $x_0, u_{1:T}$?
- **Variational EM**:
  - **E step:** Approximate the posterior with,

$$q(x_0, u_{1:T}) \approx p(x_0, u_{1:T} \mid y_{1:T}, \Theta)$$

  - **M step:** Find parameters that maximize the ELBO

$$\mathscr{L}[q, \Theta] = \mathbb{E}_{q(x_0, u_{1:T})} \left[\log p(x_0, u_{1:T}, y_{1:T}) - \log q(x_0, u_{1:T}) \right]$$



Stochastic RNNs

LFADS learning and inference: Take #1

- Let's try coordinate ascent variational inference (CAVI) with a mean field posterior approximation,

$$q(x_0, u_{1:T}) = q(x_0) \prod_{t=1}^{T} q(u_t) \approx p(x_0, u_{1:T} \mid y_{1:T}, \Theta)$$

- The optimal updates are,

$$q(u_t) \propto \exp \left\{ \mathbb{E}_{q(x_0, u_{\neg t})} \left[\log p(x_0, u_{1:T}, y_{1:T}) \right] \right\}$$

- Can we compute this expectation?

Stochastic RNNs

LFADS learning and inference: Take #1

$$q(u_t) \propto \exp\left\{ \mathbb{E}_{q(x_0, u_{\neg t})} \left[\log p(x_0, u_{1:T}, y_{1:T}) \right] \right\}$$

Stochastic RNNs

LFADS learning and inference: Take #2

- Let's assume a Gaussian form for each factor,

$$q(x_0, u_{1:T}; \lambda) = \mathcal{N}(x_0 \mid \tilde{\mu}_0, \tilde{\Sigma}_0) \prod_{t=1}^{T} \mathcal{N}(u_t \mid \tilde{\mu}_t, \tilde{\Sigma}_t)$$

- This approximation is parameterized by **variational parameters** $\lambda \triangleq \{\tilde{\mu}_t, \tilde{\Sigma}_t\}_{t=0}^{T}$.

- Let $\mathscr{L}(\lambda, \Theta) = \mathscr{L}[q(x_0, u_{1:T}; \lambda), \Theta]$ denote the ELBO as a function of the variational and generative model parameters.

Stochastic RNNs

LFADS learning and inference: Take #2

ELBO Surgery*: we can rewrite the ELBO as,

$$\begin{aligned}
\mathscr{L}(\lambda, \Theta) &= \mathbb{E}_{q(x_0, u_{1:T}, \lambda)} \left[\log p(x_0, u_{1:T}) + \log p(y_{1:T} \mid x_0, u_{1:T}, \Theta) - \log q(x_0, u_{1:T}; \lambda) \right] \\
&= \mathbb{E}_{q(x_0, u_{1:T}, \lambda)} \left[\log p(y_{1:T} \mid x_0, u_{1:T}, \Theta) - \log \frac{q(x_0; \lambda)}{p(x_0)} - \sum_{t=1}^{T} \log \frac{q(u_t; \lambda)}{p(u_t)} \right] \\
&= \underbrace{\mathbb{E}_{q(x_0, u_{1:T}, \lambda)} \left[\sum_{t=1}^{T} \log p(y_t \mid x_0, u_{1:t}, \Theta) \right]}_{\text{expected log likelihood}} \underbrace{- \mathrm{KL}(q(x_0; \lambda) \,\|\, p(x_0)) - \sum_{t=1}^{T} \mathrm{KL}(q(u_t; \lambda) \,\|\, p(u_t))}_{\text{KL to the prior}}
\end{aligned}$$

*For more ways of rewriting the ELBO, see Johnson and Hoffman (2017)

Stochastic RNNs

LFADS learning and inference: gradients wrt Θ

Gradient ascent on the ELBO:

$$\nabla_\Theta \mathscr{L}(\lambda, \Theta) = \mathbb{E}_{q(x_0, u_{1:T}, \lambda)} \left[\sum_{t=1}^{T} \nabla_\Theta \log p(y_t \mid x_0, u_{1:t}, \Theta) \right]$$

Since the generative parameters don't appear in q, we can **pull the gradient inside the expectation** and compute it with **automatic differentiation** for any $x_0, u_{1:t}, \Theta$.

Then approximate the expectation with **Monte Carlo:**

$$\nabla_\Theta \mathscr{L}(\lambda, \Theta) \approx \frac{1}{S} \sum_{s=1}^{S} \left[\sum_{t=1}^{T} \nabla_\Theta \log p(y_t \mid x_0^{(s)}, u_{1:t}^{(s)}, \Theta) \right] \qquad x_0^{(s)} \sim q(x_0; \lambda),\ u_t^{(s)} \sim q(u_t; \lambda).$$

Stochastic RNNs

LFADS learning and inference: the “reparameterization trick”

The gradients with respect to the variational parameters are a bit trickier:

$$\nabla_\lambda \mathscr{L}(\lambda, \Theta) = \nabla_\lambda \mathbb{E}_{q(x_0, u_{1:T}, \lambda)} \left[\sum_{t=1}^{T} \log p(y_t \mid x_0, u_{1:t}, \Theta) \right] - \nabla_\lambda \mathrm{KL}\Big(q(x_0, u_{1:T}, \lambda) \,\|\, p(x_0, u_{1:T}) \Big)$$

Note that $x_0 \sim \mathcal{N}(\tilde{\mu}_0, \tilde{\Sigma}_0) \iff x_0 = \tilde{\mu}_0 + \tilde{\Sigma}_0^{1/2} \epsilon_0$ where $\epsilon_0 \sim \mathcal{N}(0, I)$.

We can **reparameterize the model** in terms of an expectation wrt $\epsilon_{0:T}$ and then take the gradient inside the expectation, as before

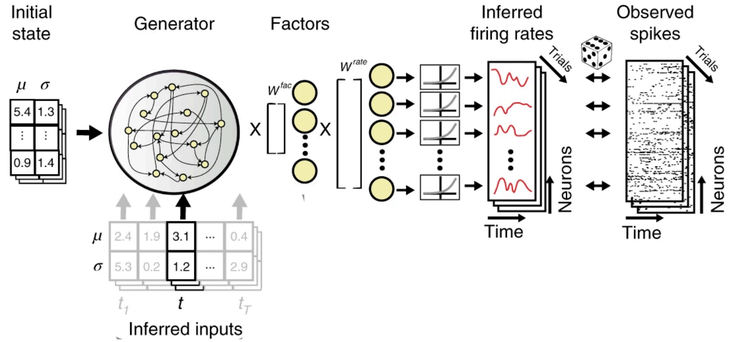
$$\nabla_\lambda \mathscr{L}(\lambda, \Theta) = \mathbb{E}_{\epsilon_{0:T}} \left[\sum_{t=1}^{T} \nabla_\lambda \log p(y_t \mid x_0(\epsilon_0, \lambda), u_1(\epsilon_1, \lambda), \ldots, u_t(\epsilon_t, \lambda), \Theta) \right] - \nabla_\lambda \mathrm{KL}\Big(q(x_0, u_{1:T}, \lambda) \,\|\, p(x_0, u_{1:T}) \Big)$$

As before, we can approximate this with ordinary Monte Carlo.

Stochastic RNNs

LFADS learning and inference

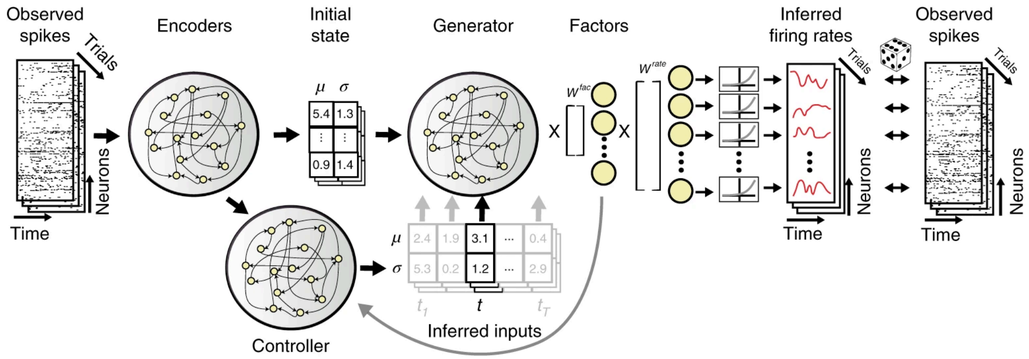
- **Variational EM** via gradient descent and the reparameterization trick,
  - **E step:**
    - Draw $\epsilon_t^{(s)} \sim \mathcal{N}(0,I)$ for $t = 0,\ldots,T$, $s = 1,\ldots,S$.
    - Use ϵ to approximate $\nabla_\lambda \mathcal{L}(\lambda, \Theta)$ via Monte Carlo and the reparameterization trick.
    - Update $\lambda \leftarrow \lambda + \alpha \nabla_\lambda \mathcal{L}(\lambda, \Theta)$
  - **M step:**
    - Use ϵ to approximate $\nabla_\Theta \mathcal{L}(\lambda, \Theta)$ via Monte Carlo.
    - Update $\Theta \leftarrow \Theta + \alpha \nabla_\Theta \mathcal{L}(\lambda, \Theta)$.



Stochastic RNNs

Amortized inference with encoders / recognition networks

- With large datasets, we often work on one mini-batch at a time.
- In that setting, we need a way to quickly obtain a decent posterior approximation for that mini-batch.
- **Key idea**: the optimal λ is a function of the data $y_{1:T}$, so let's **use a neural network** to approximate the mapping from data to variational parameters.
- This is called **amortized inference.**
- The learned network is called an **encoder** or a **recognition network**.



Conclusion

- We can generalize the SLDS in a number of ways while retaining nice conditional structure.
- We can only get so far with closed form updates and coordinate ascent. For example, Stochastic RNNs are not generally amenable to simple updates.
- In such cases, we can use stochastic gradient ascent on the ELBO with a handful of tricks: parametric assumptions, Monte Carlo approximation, the reparameterization trick…
- LFADS is one example of where such methods have been used to achieve state of the art performance in a variety of neuroscience tasks.

Further Reading

- Pandarinath, Chethan, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, et al. 2018. "Inferring Single-Trial Neural Population Dynamics Using Sequential Auto-Encoders." Nature Methods 15 (10): 805–15.

- Linderman, Scott W., Matthew J. Johnson, Andrew C. Miller, Ryan P. Adams, David M. Blei, and Liam Paninski. 2017. "Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems." In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS).

- Nassar, Josue, Scott W. Linderman, Monica Bugallo, and Il Memming Park. 2019. "Tree-Structured Recurrent Switching Linear Dynamical Systems for Multi-Scale Modeling." In International Conference on Learning Representations (ICLR). http://arxiv.org/abs/https://.org/abs/1811.12386.

- Linderman, Scott W., Annika Nichols, David M. Blei, Manuel Zimmer, and Liam Paninski. 2018. "Hierarchical Recurrent Models Reveal Latent States of Neural Activity in C. Elegans." Computational and Systems Neuroscience (Cosyne) Abstracts.

- Zoltowski, David, Jonathan Pillow, and Scott Linderman. 2020. "A General Recurrent State Space Framework for Modeling Neural Dynamics during Decision-Making." In Proceedings of the 37th International Conference on Machine Learning, edited by Hal Daumé Iii and Aarti Singh, 119:11680–91. Proceedings of Machine Learning Research. PMLR.