Machine Learning Methods for Neural Data Analysis

Lecture 11: Unsupervised modeling

Scott Linderman

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

Announcements

- **Proposals** look great!
  - Feedback coming in the next day or two.
  - Next update due Mar 5 (next Friday).

Agenda

Finish decoding and start unsupervised modeling

- “Direct” decoders and structured prediction
- Unit III: Unsupervised models of neural and behavioral data
- Bayesian inference in latent variable models

Decoding movement from neural spike trains

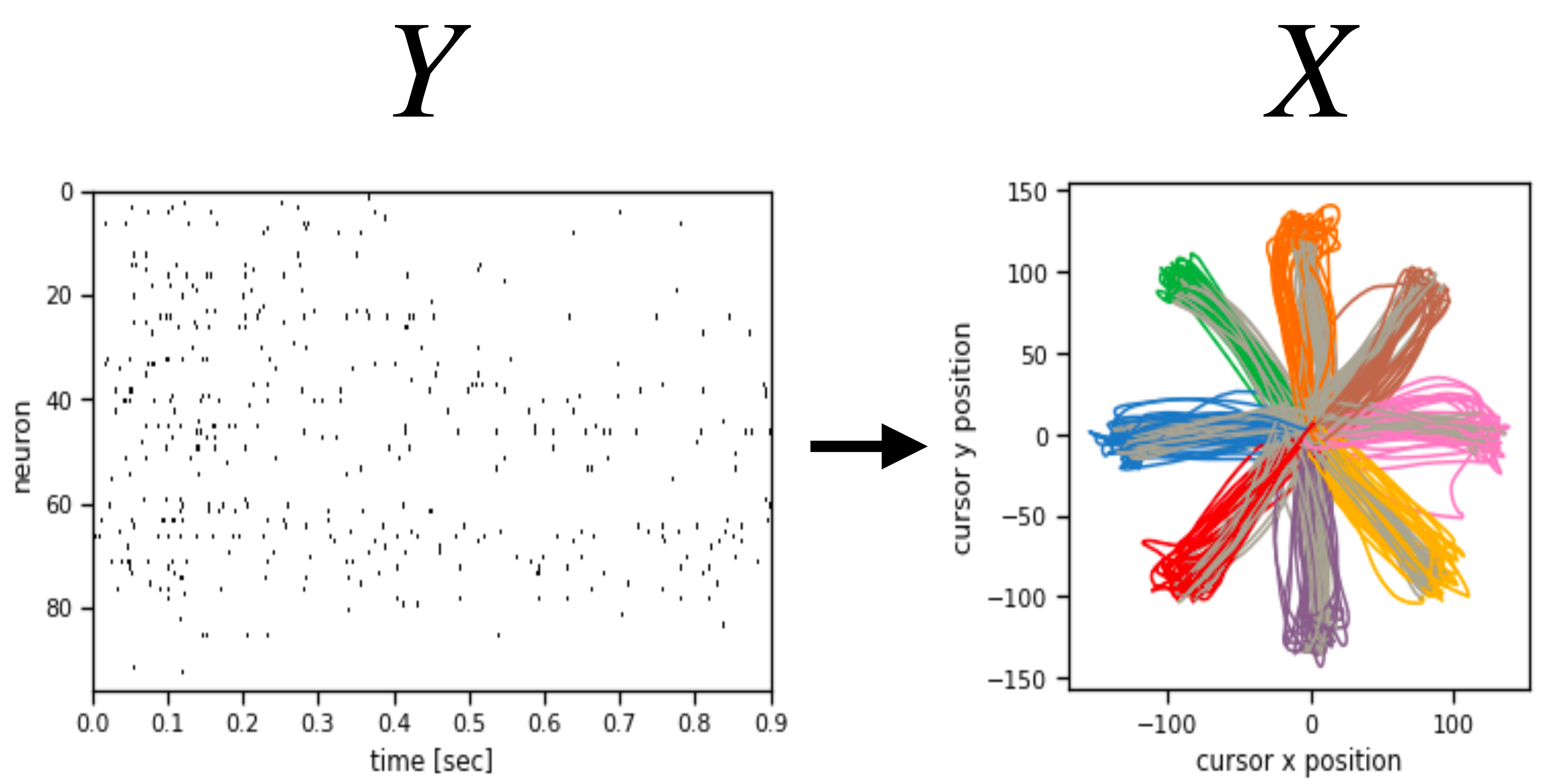
A linear dynamical system (LDS) model

- One of the problems with the basic model (Lab 6 Part 1) is that it treated each time bin as independent.
- Instead, consider the following prior

$$p(X) = p(x_1)\prod_{t=2}^{T} p(x_t \mid x_{t-1})$$

$$= \mathcal{N}(x_1 \mid 0, Q)\prod_{t=2}^{T} \mathcal{N}(x_t \mid Ax_{t-1}, Q)$$

- Parameterized by **dynamics matrix** $A \in \mathbb{R}^{D\times D}$ and **dynamics covariance** $Q \in \mathbb{R}^{D\times D}$.



Decoding movement from neural spike trains

The posterior distribution

The posterior is given by

$$
\begin{aligned}
p(X \mid Y) &\propto \left[\mathcal{N}(x_1 \mid 0, Q) \prod_{t=2}^{T} \mathcal{N}(x_t \mid A x_{t-1}, Q)\right] \left[\prod_{t=1}^{T} \mathcal{N}(y_t \mid C x_t + d, R)\right] \\
&= \exp\{\text{"a big quadratic function of } X\text{"}\} \\
&= \mathcal{N}(\mathrm{vec}(X) \mid \mu, \Sigma)
\end{aligned}
$$

Where $\Sigma = J^{-1}$ and $\mu = J^{-1}h$ with,

$$
J = \begin{bmatrix} J_{11} & J_{21}^\top & & & \\ J_{21} & J_{22} & J_{32}^\top & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & J_{T,T-1}^\top \\ & & & J_{T,T-1} & J_{TT} \end{bmatrix} \qquad h = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_T \end{bmatrix}
$$

- The blocks are given by $J_{tt} = Q^{-1} + A^\top Q^{-1} A + C^\top R^{-1} C$ (except for J_{11} and J_{TT}), $J_{t,t-1} = -Q^{-1}A$, and $h_t = C^\top R^{-1}(y_t - d)$.

Decoding movement from neural spike trains

Poisson observations

- So far we've used a linear, Gaussian encoder for the spikes, even though they are counts!
- Suppose instead,

$$p(Y \mid X) = \prod_{t=1}^{T} \prod_{n=1}^{N} \mathrm{Po}\left(y_{tn} \mid f(c_n^\top x_t + d_n)\right)$$

- The posterior is no longer Gaussian, but it's common to approximate it as one.

Decoding movement from neural spike trains

Laplace approximation

Approximate the posterior as

$$p(X \mid Y) \approx \mathcal{N}(\mu, \Sigma)$$

where

$$\begin{aligned}
\mathscr{L}(X) &= -\log p(X, Y) \\
\mu &= \operatorname{argmin}_X \mathscr{L}(X) \\
\Sigma &= \left[\nabla^2 \mathscr{L}(X) \Big|_{X=\mu} \right]^{-1}
\end{aligned}$$

For GLM encoders, the log joint is concave and μ and Σ can be found efficiently.

Decoding movement from neural spike trains

Laplace approximation under a Poisson GLM encoder

Derive the Hessian under the Poisson GLM encoder with exponential mean function $f(a) = e^a$:

$$
\begin{aligned}
\frac{\partial^2}{\partial x_t \partial x_t} \mathscr{L}(X) &= J_{tt} - \sum_{n=1}^{N} \frac{\partial^2}{\partial x_t \partial x_t} \log \mathrm{Po} \left(y_{tn} \mid f(c_n^\top x_t + d_n) \right) \\
&= J_{tt} - \sum_{n=1}^{N} \frac{\partial^2}{\partial x_t \partial x_t} \left[-f(c_n^\top x_t + d_n) + y_{tn} \log f(c_n^\top x_t + d_n) \right] \\
&= J_{tt} + \sum_{n=1}^{N} \exp\{c_n^\top x_t + d_n\} c_n c_n^\top
\end{aligned}
$$

Decoding movement from neural spike trains

Structured decoders

- If we're going to make a Gaussian approximation anyway, why not learn more flexible means and covariances?

- Recall the form of the LDS posterior,

$$J_{tt} = Q^{-1} + A^\top Q^{-1} A + C^\top R^{-1} C$$

$$J_{t,t-1} = - Q^{-1} A$$

$$h_t = C^\top R^{-1} (y_t - d)$$

- **Idea**: replace these with learned functions of $y_{1:T}$.

Decoding movement from neural spike trains

Structured decoders

For example,

$$p(X \mid Y) = \mathcal{N}(\mathrm{vec}(X) \mid \mu, \Sigma)$$

$$\mu = J(Y)^{-1} h(Y)$$

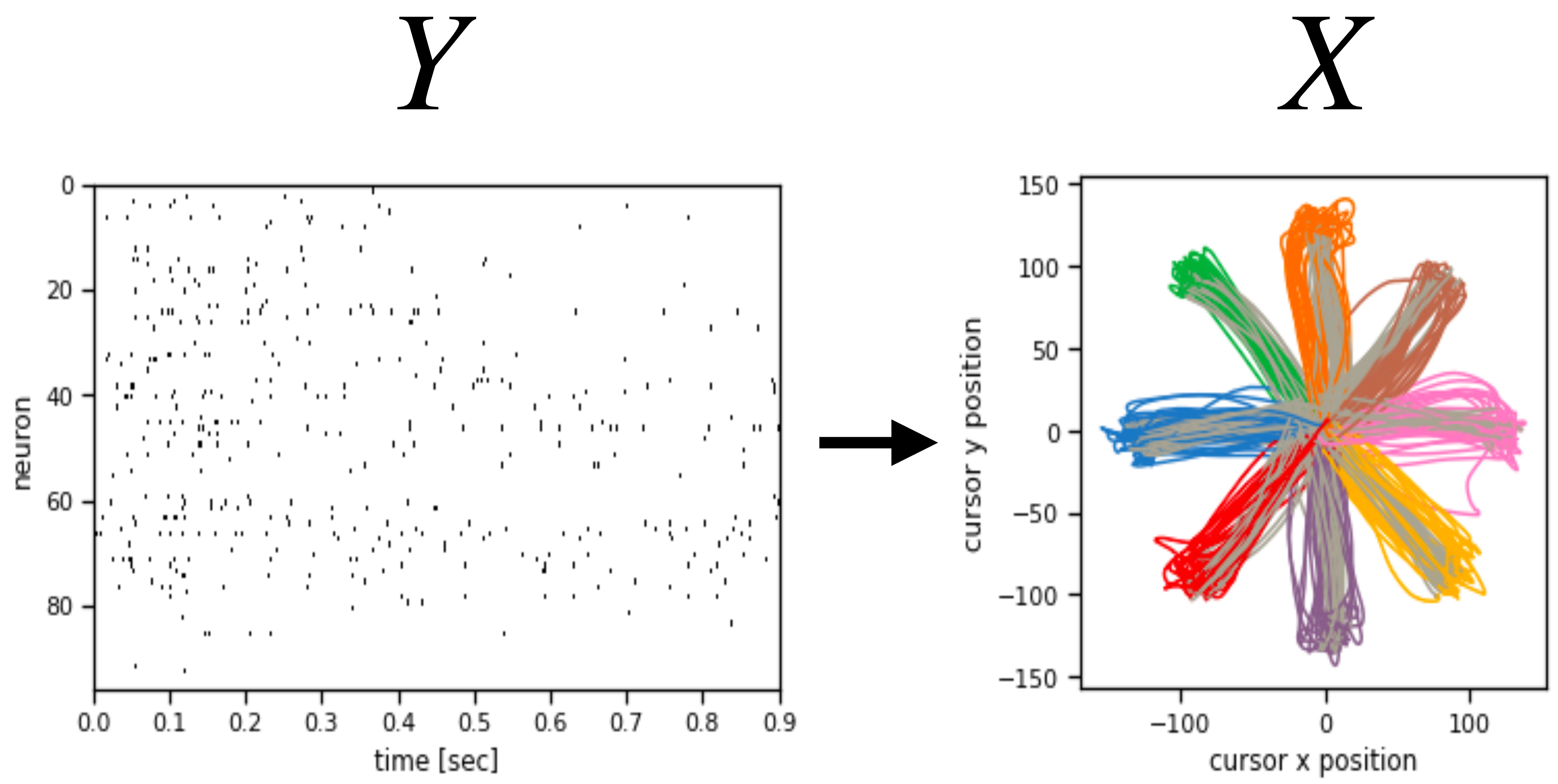
$$\Sigma = J(Y)^{-1}$$

Where

$$J_{tt} = Q^{-1} + A^\top Q^{-1} A + f(y_{t-\Delta:t+\Delta})$$

$$J_{t,t-1} = -Q^{-1} A$$

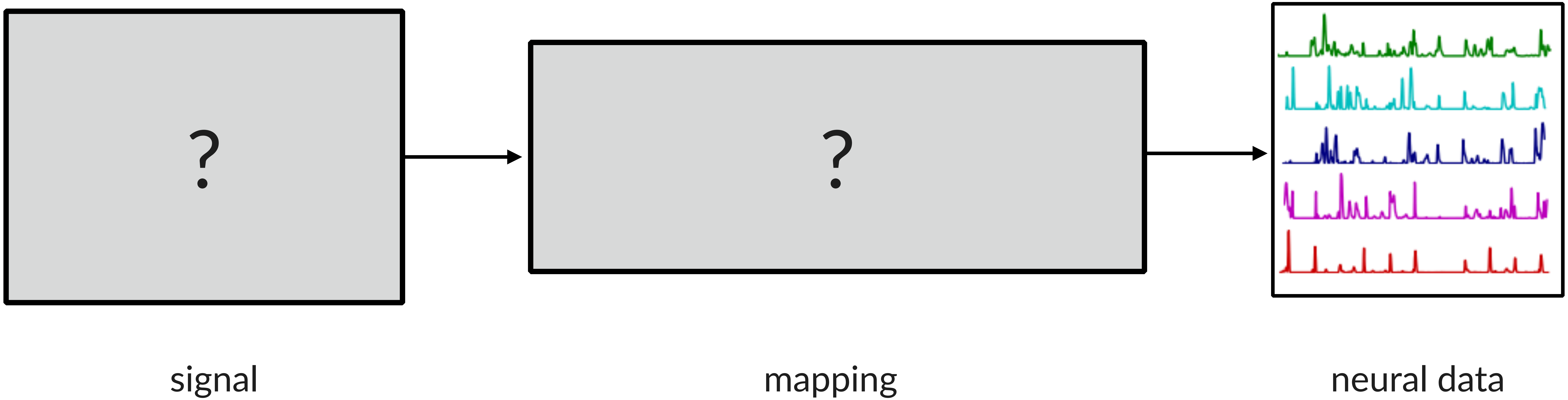
$$h_t = g(y_{t-\Delta:t+\Delta})$$



Unit III: Unsupervised modeling

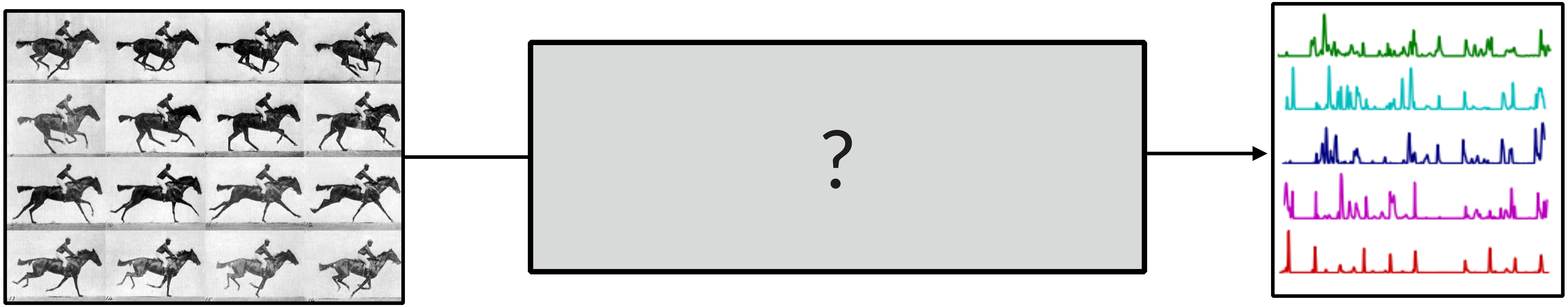
Data-driven modeling

Searching for signals to explain neural activity



Data-driven modeling

Searching for signals to explain neural activity



signal

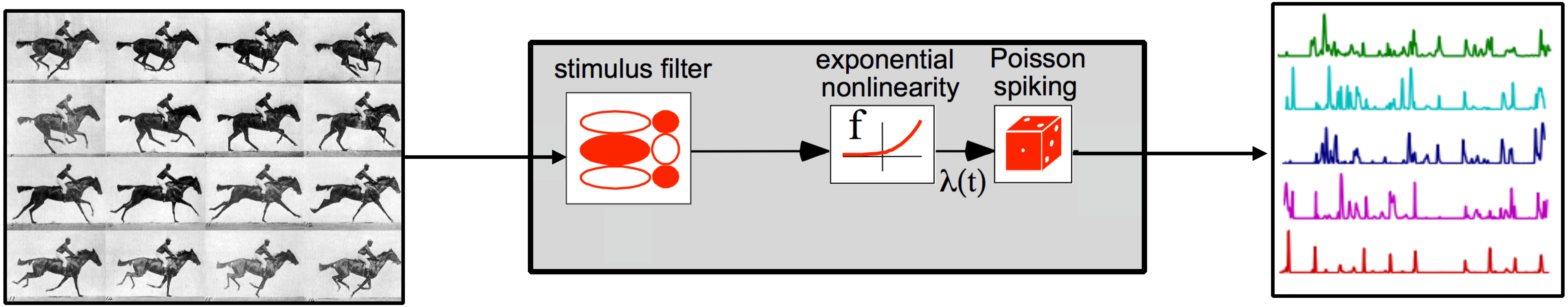
mapping

neural data

Encoding models: given stimulus (covariates) and response, find mapping.

Data-driven modeling

Searching for signals to explain neural activity



signal

mapping

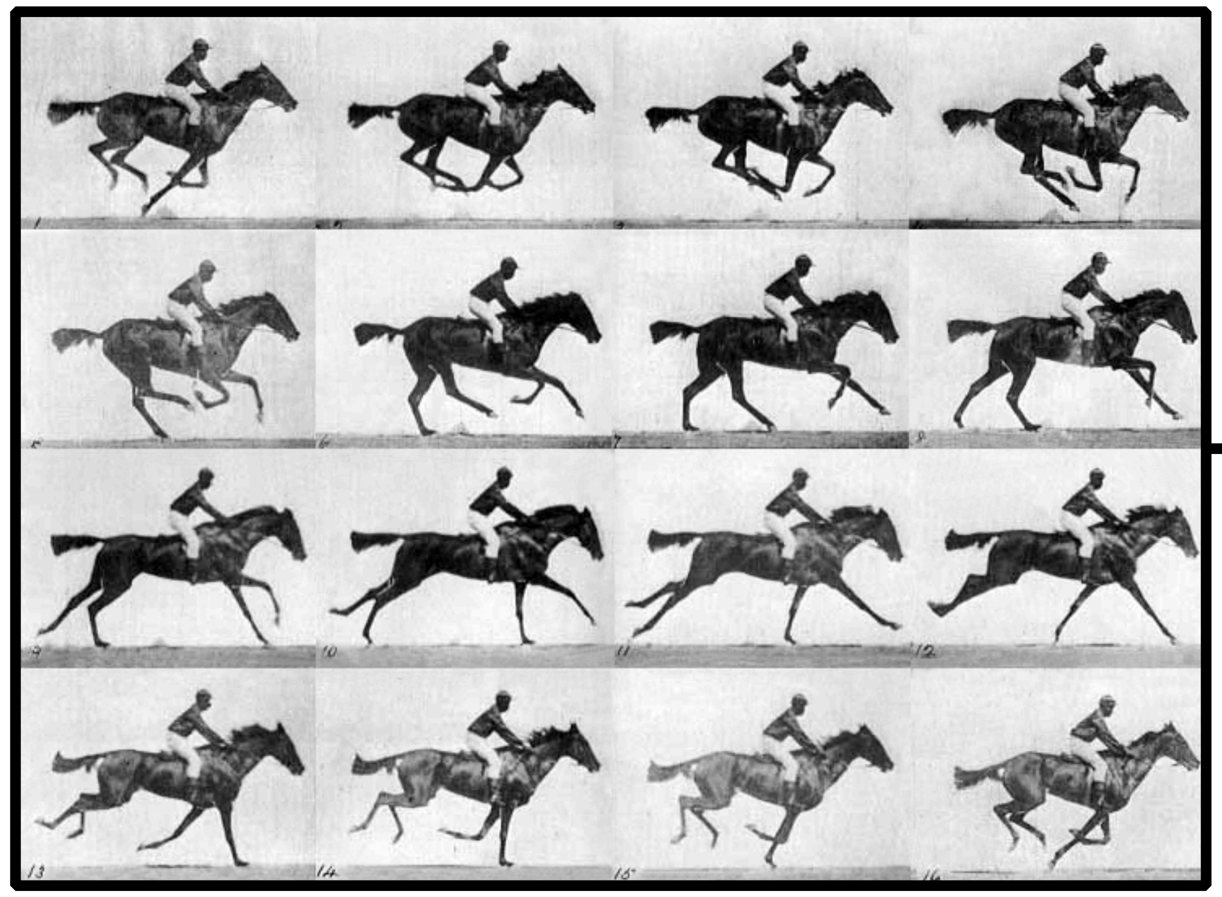
neural data

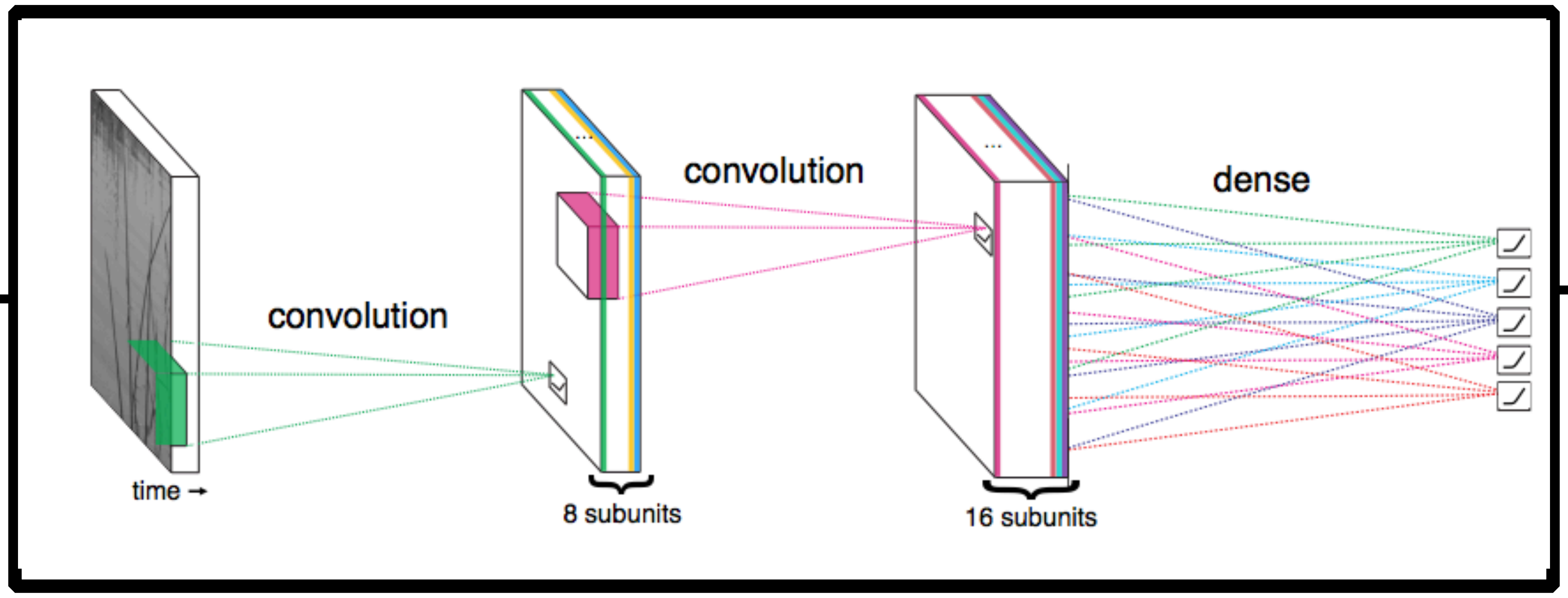
Recent examples: Musall et al (2018), Stringer et al (2018)

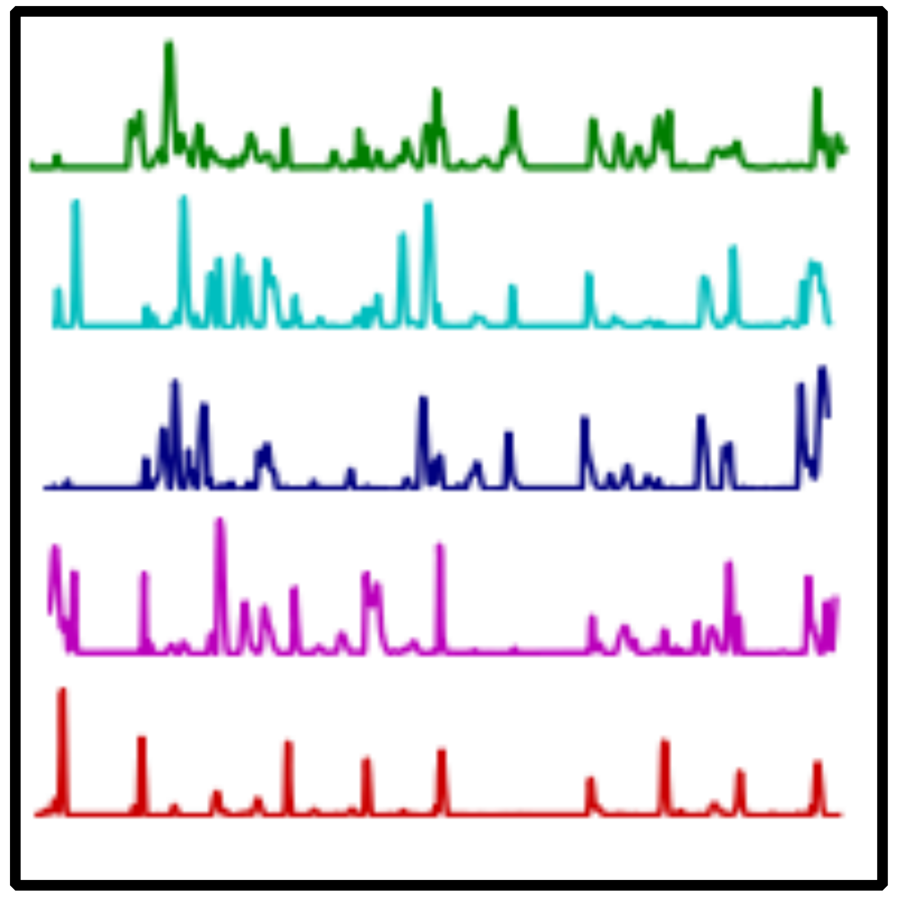
Paninski (2004)
Truccolo et al (2005)
Pillow et al (2008)

Data-driven modeling

Searching for signals to explain neural activity







signal

mapping

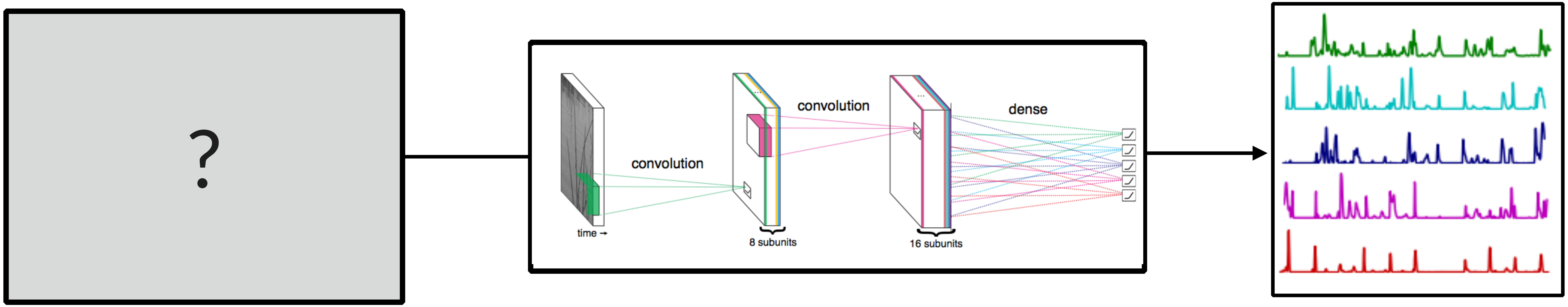
neural data

Toward nonlinear and/or more biophysically plausible mappings.

McIntosh et al (2017)

Data-driven modeling

Searching for signals to explain neural activity



latent signal

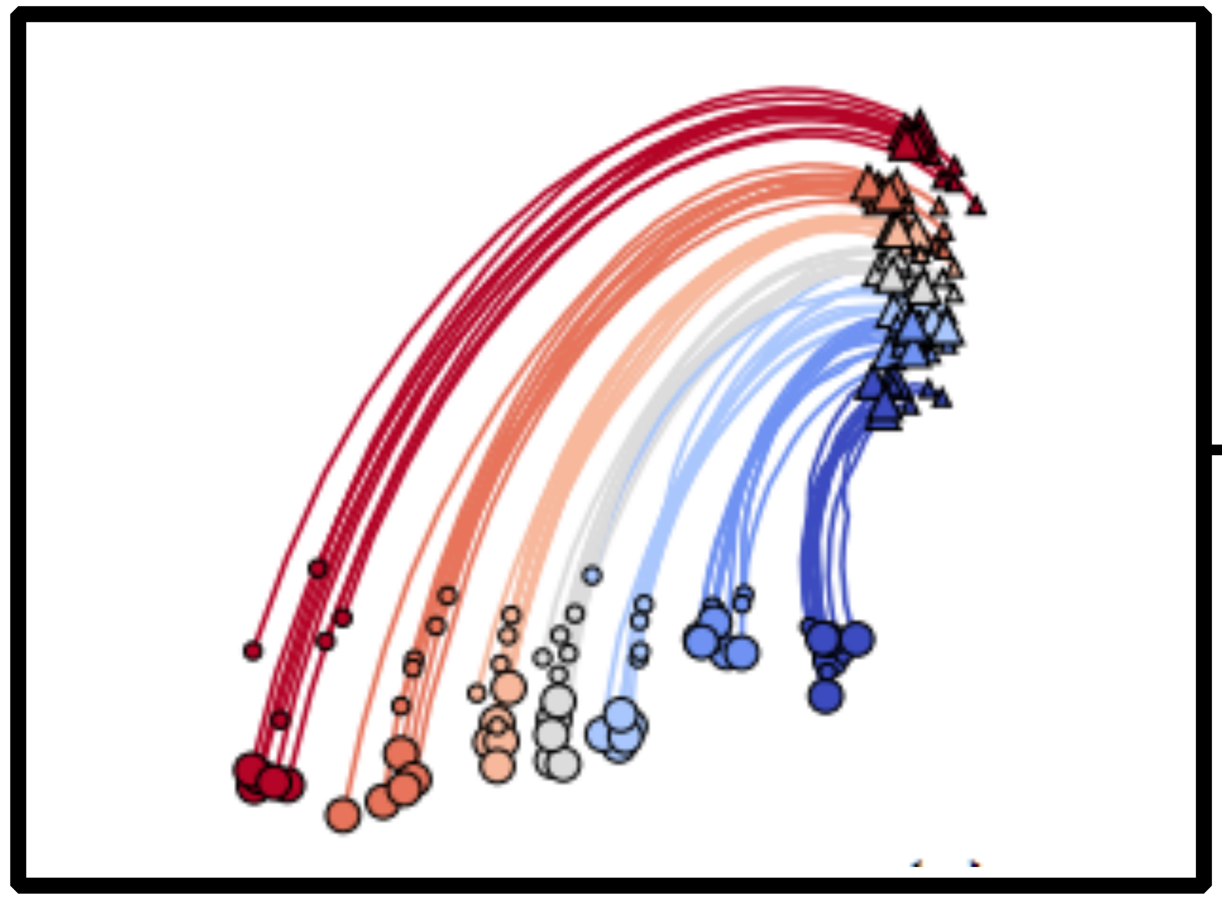
mapping

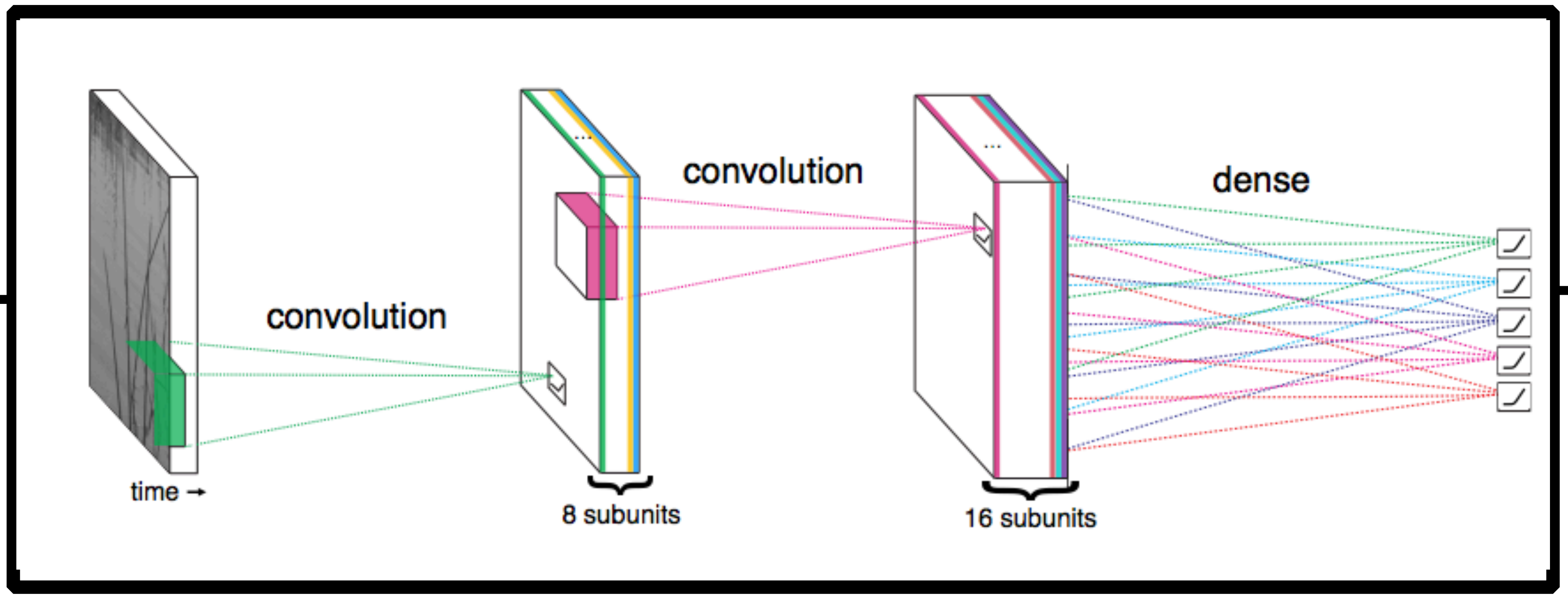
neural data

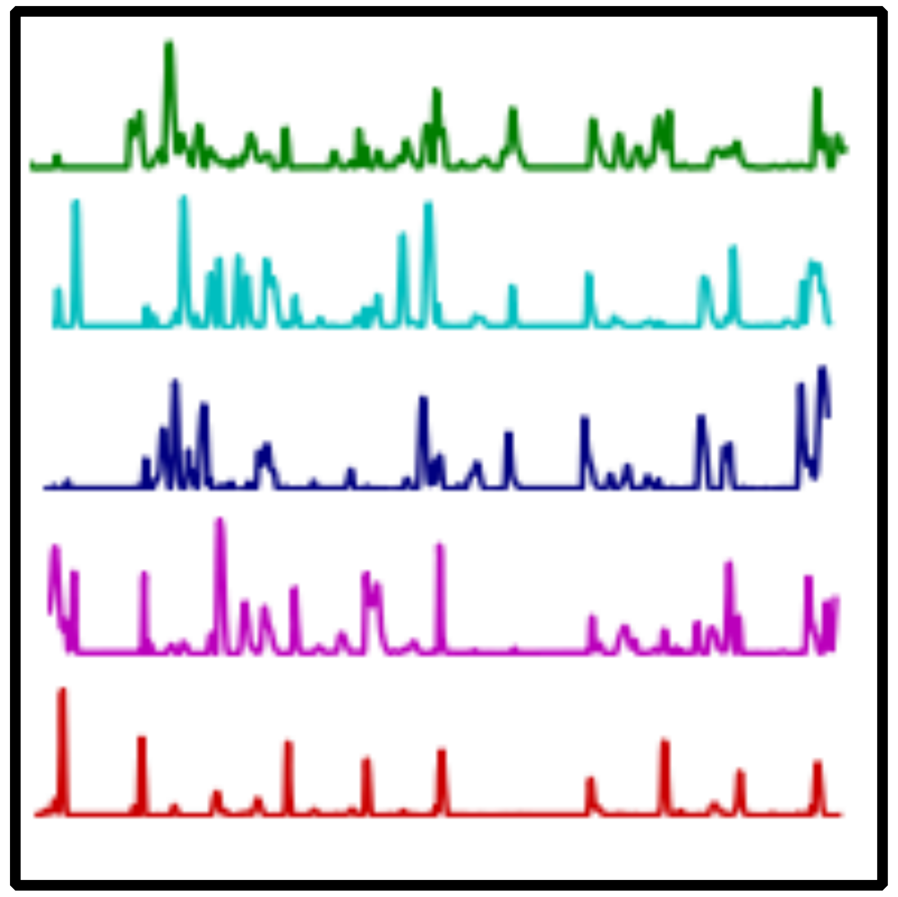
Alternative: try to infer latent signals from the data

Data-driven modeling

Searching for signals to explain neural activity







latent signal

mapping

neural data

Alternative: try to infer latent signals from the data, *subject to constraints.*

Latent variable modeling is all about constraints

The five D's

- *Dimensionality*: how many latent clusters, factors, etc.?
- *Domain*: are the latent variables discrete, continuous, bounded, sparse, etc.?
- *Dynamics*: how do the latent variables change over time?
- *Dependencies*: how do the latent variables relate to the observed data?
- *Distribution*: do we have prior knowledge about the variables' probability?

- We've already seen some examples in Unit 1!

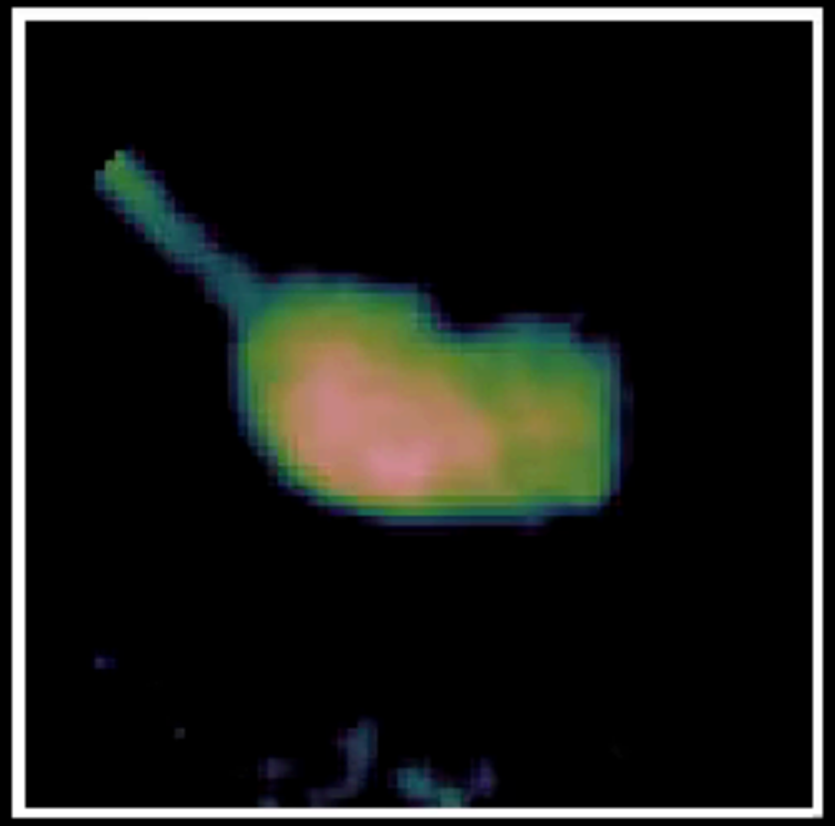
Latent variable modeling is all about constraints

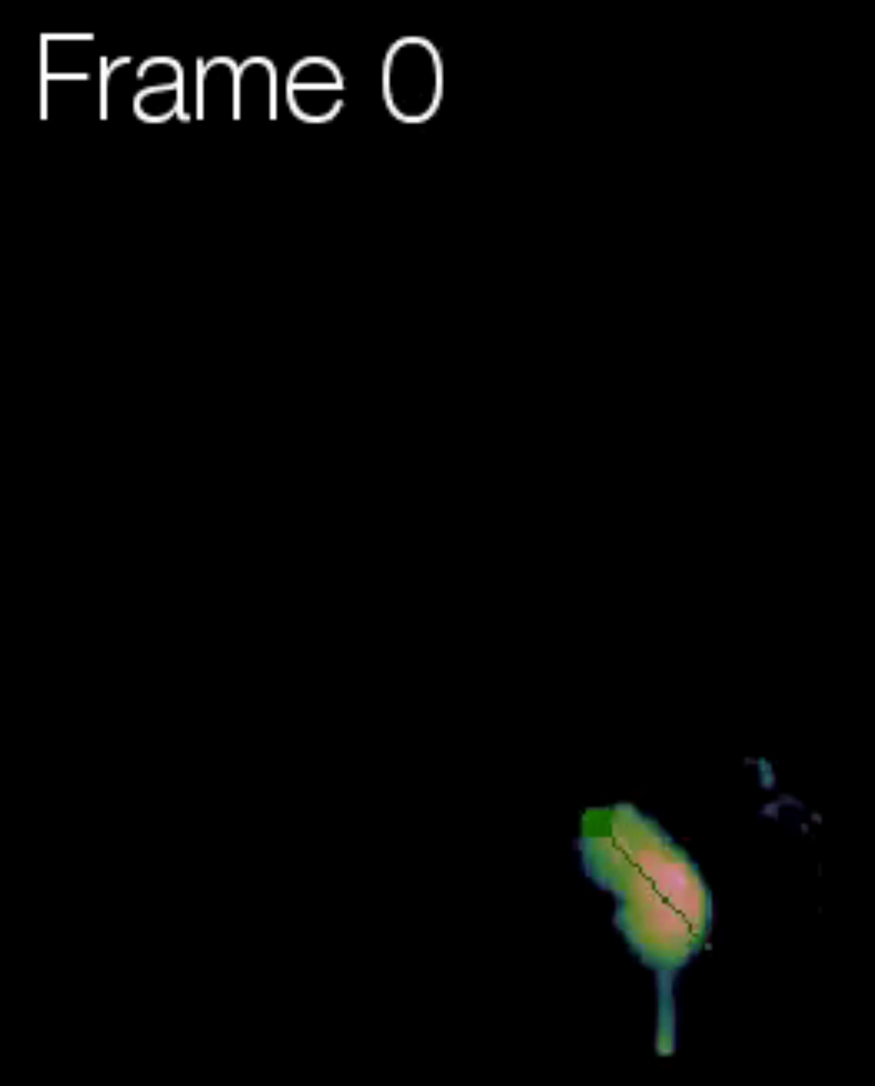
Domain/Dependency/Distribution

Dynamics /Domain

	Continuous Linear Gaussian	Discrete (Gen.) Linear Bernoulli/Poisson/etc.	Nonlinear Observation Models
Discrete Markovian Categorical	**HMM** *Rabiner (1989)*	**HMM** *Rabiner (1989)*	**Structured VAE** *Johnson et al (2016)*
Continuous Linear Gaussian	**LDS** *Kalman (1960)*	**Poisson LDS** *Smith and Brown (2003), Paninski et al (2010) Macke et al (2011)*	**Deep PfLDS** Archer et al (2015); Gao et al (2016)
Continuous Nonlinear (parametric) Gaussian	**NLDS, e.g. Hodgkin-Huxley** *Ahrens, Huys, Paninski (2006) Huys and Paninski (2009)*	**NLDS, e.g. Hodgkin-Huxley** *Meng, Kramer, Eden (2011)*	**GPSSM, DKF, LFADS, VIND** *Frigola et al (2013) , Krishnan et al (2015), Sussillo et al (2016), Hernandez et al (2018)*
Mixed Switching Linear	**SLDS** *Ghahramani and Hinton (1996) Murphy (1998)*	**Poisson SLDS** *Petreska et al (2013)*	**Structured VAE** *Johnson et al (2016)*
Mixed Recurrent Linear	**recurrent/augmented SLDS** *Barber (2006); Pachitariu et al (2014); Linderman et al (2017); Nassar et al (2019)*	**rSLDS** *Linderman et al (2017) Nassar et al (2019)*	**Structured VAE** *Johnson et al (2016)*
Continuous Nonlinear (smoothing) Gaussian	**GPFA** *Yu, Cunningham, et al (2009)*	**vLGP** *Zhao and Park (2017)*	**GPLVM** *Lawerence (2005), Wu et al (2017)*
Continuous Nonlinear (nonparametric) Gaussian	**GPSSM, DKF, LFADS, VIND** *Frigola et al (2013) , Krishnan et al (2015), Sussillo et al (2016), Hernandez et al (2018)*	**GPSSM, DKF, LFADS, VIND** *Frigola et al (2013) , Krishnan et al (2015), Sussillo et al (2016), Hernandez et al (2018)*	**GPSSM, DKF, LFADS, VIND** *Frigola et al (2013) , Krishnan et al (2015), Sussillo et al (2016), Hernandez et al (2018)*

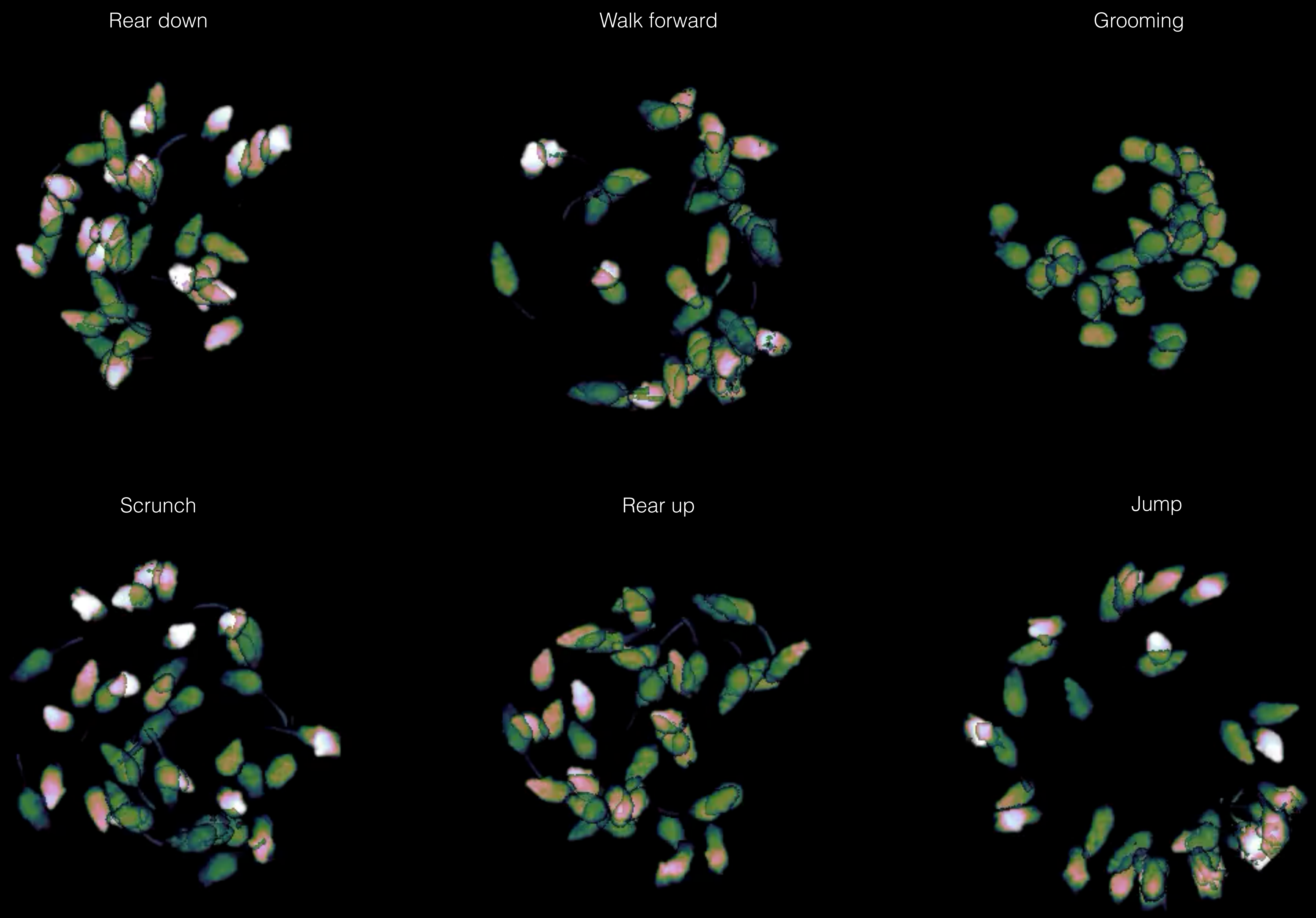
Motivating Example: summarizing videos with behavioral states





Wiltschko et al, 2015

Motivating Example: summarizing videos with behavioral states



Wiltschko et al, 2015

Formulating as a probabilistic model

- **Variables:** Let,
  - $y_t \in \mathbb{R}^P$ denote the (vectorized) image at time t.
  - $z_t \in \{1,\ldots,K\}$ denote the discrete latent state (aka behavioral "syllable") at time t.
- **Model:** Assume each time frame is independent and,

$$\begin{aligned} z_t &\sim \mathrm{Cat}(\pi) \\ y_t \mid z_t &\sim \mathcal{N}(d_{z_t}, R_{z_t}) \end{aligned}$$

- **Parameters:** Let $\Theta = \pi, \{d_k, R_k\}_{k=1}^K$ denote the parameters,
  - $\pi \in \Delta_K$ is the prior probability of each state
  - $(d_k, R_k) \in \mathbb{R}^P \times \mathbb{R}^{P\times P}$ are the conditional mean and variance of images for discrete state $z_t = k$.

Bayesian inference in latent variable models

Bayesian inference in latent variable models

MAP Estimation

- In Unit 1 we used ***maximum a posteriori*** **(MAP) estimation** to find,

$$z^\star, \Theta^\star = \arg\max_{z,\Theta} \log p(y, z, \Theta)$$

- This gave us a **point estimate** of the latent variables z and parameters Θ.
- Point estimates can lead to an **overly optimistic** view of the model.
- Specifically, MAP estimation found **the best assignment**, which may not reflect the **average performance** under the prior $p(z, \Theta)$.

Bayesian inference in latent variable models

Integrating over the latent variables

- A more **Bayesian approach** is to **integrate** over the latent variables.
- First, **learn** a point estimate of the parameters,

$$\Theta^\star = \arg\max_\Theta \log p(y, \Theta)$$

where $p(y, \Theta) = \int p(y, z, \Theta)\, \mathrm{d}z = \mathbb{E}_{p(z,\Theta)}[p(y \mid z, \Theta)]$ is the **marginal likelihood.**

- Then, **infer** the posterior distribution over latent variables given observed data and parameters,

$$p(z \mid y, \Theta) = \frac{p(y \mid z, \Theta)\, p(z \mid \Theta)\, p(\Theta)}{p(y, \Theta)}$$

- (A “fully Bayesian” approach would integrate over both z and Θ.)

Bayesian inference in latent variable models

Maximizing the marginal likelihood

- How to learn the parameters?
- First idea: **gradient ascent**,

$$\nabla_\Theta \log p(y, \Theta) = \frac{\nabla_\Theta p(y, \Theta)}{p(y, \Theta)} = \frac{\int \nabla_\Theta p(y, z, \Theta)\, \mathrm{d}z}{\int p(y, z, \Theta)\, \mathrm{d}z}$$

- Sometimes, these integrals are available in **closed form**.
  - For example, when z **is discrete** the integrals become sums.
- Can we do better?

Bayesian inference in latent variable models

Lower bound the marginal likelihood

- Next idea: lower bound the marginal likelihood with a more tractable form,

$$\log p(y, \Theta) = \log \int p(y, z, \Theta) \, \mathrm{d}z$$

Bayesian inference in latent variable models

Lower bound the marginal likelihood

- Next idea: lower bound the marginal likelihood with a more tractable form,

$$
\begin{aligned}
\log p(y, \Theta) &= \log \int p(y, z, \Theta) \, \mathrm{d}z \\
&= \log \int \frac{q(z)}{q(z)} p(y, z, \Theta) \, \mathrm{d}z && \text{for any distribution } q(z) \\
&= \log \mathbb{E}_{q(z)} \left[\frac{p(y, z, \Theta)}{q(z)} \right] \\
&\geq \mathbb{E}_{q(z)} \left[\log p(y, z, \Theta) - \log q(z) \right] && \text{by Jensen's inequality} \\
&\triangleq \mathscr{L}[q, \Theta]
\end{aligned}
$$

- $\mathscr{L}$ is called the **evidence lower bound** or the **ELBO** for short.

Bayesian inference in latent variable models

Coordinate ascent on the ELBO

- Update the parameters,

$$\Theta \leftarrow \arg\max_{\Theta} \mathscr{L}[q, \Theta] = \arg\max_{\Theta} \mathbb{E}_{q(z)}[\log p(y, z, \Theta)]$$

- Update the distribution on latent variables,

$$\begin{aligned} q &\leftarrow \arg\max_{q} \mathscr{L}[q, \Theta] \\ &= \arg\max_{q} \mathbb{E}_{q(z)}\left[\frac{\log p(y, z, \Theta)}{q(z)}\right] \\ &= \arg\min_{q} \mathrm{KL}\left(q(z) \,\|\, p(z \mid y, \Theta)\right) \\ &= p(z \mid y, \Theta) \end{aligned}$$

Bayesian inference in latent variable models

The Expectation-Maximization (EM) algorithm

- **M-step**: Maximize the expected log probability

$$\Theta \leftarrow = \arg\max_{\Theta} \mathbb{E}_{q(z)}[\log p(y, z, \Theta)]$$

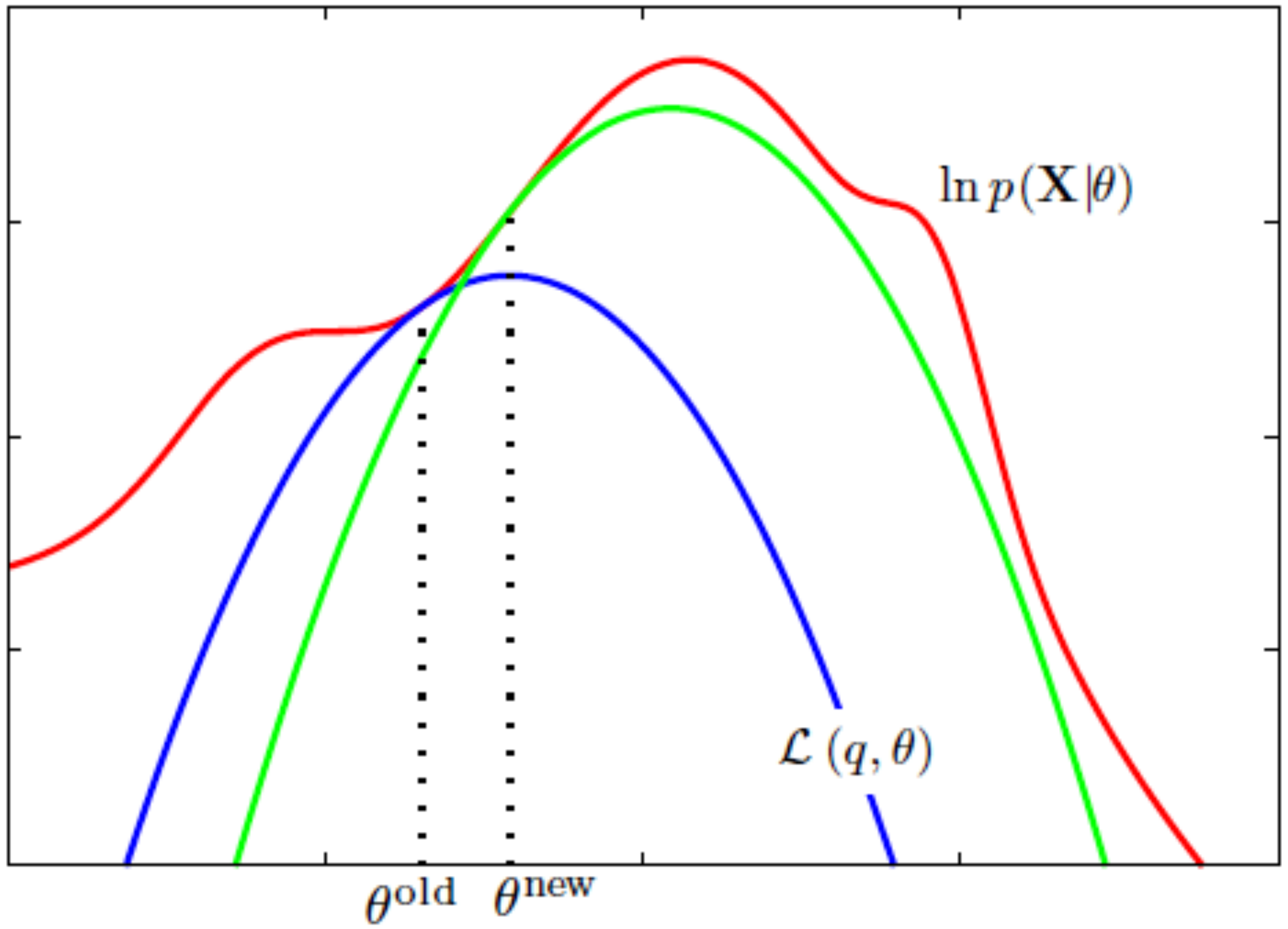
- **E-step**: Update the posterior over latent variables

$$q \leftarrow p(z \mid y, \Theta)$$

- After each E-step, the **ELBO is tight**:

$$\begin{aligned}
\mathscr{L}[p(z \mid y, \Theta), \Theta] &= \mathbb{E}_{p(z|y,\Theta)}\left[\log \frac{p(y, z, \Theta)}{p(z \mid y, \Theta)}\right] \\
&= \mathbb{E}_{p(z|y,\Theta)}\left[\log p(y, \Theta)\right] \\
&= \log p(y, \Theta)
\end{aligned}$$

- EM converges to **local optima** of the marginal distribution.



Bishop (2006). Pattern Recognition and Machine Learning, Ch 9.4.

Bayesian inference in latent variable models

EM for the Gaussian mixture model

Recall the model,

$$z_t \sim \mathrm{Cat}(\pi)$$
$$y_t \mid z_t \sim \mathcal{N}(d_{z_t}, R_{z_t})$$

with parameters $\Theta = \pi, \{d_k, R_k\}_{k=1}^K$.

- **E-step**: Update the posterior over latent variables,

$$q(z_t = k) \leftarrow p(z_t = k \mid y_t, \Theta) \propto \frac{\pi_k \mathcal{N}(y_T \mid d_k, R_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(y_t \mid d_j, R_j)}$$

- **M-step**: Update the parameters. Let $N_k = \sum_{t=1}^T q(z_t = k)$, then

$$\pi_k \leftarrow \frac{N_k}{T}$$
$$d_k \leftarrow \frac{1}{N_k} \sum_{t=1}^T q(z_t = k)\, y_t$$
$$R_k \leftarrow \frac{1}{N_k} \sum_{t=1}^T q(z_t = k)\, (y_t - d_k)(y_t - d_k)^\top$$

Conclusion

- Unsupervised models, specifically **latent variable models,** seek simple underlying variables to explain neural or behavioral data.
- LVMs are all about **constraints.**
- MAP estimation, which we used in Unit 1, yielded a point estimate, but can be over-optimistic.
- EM maximizes the marginal likelihood by coordinate ascent on the **latent variable posterior** and the **parameters.**