Latent Variable Models

Overview of Latent Variable Models

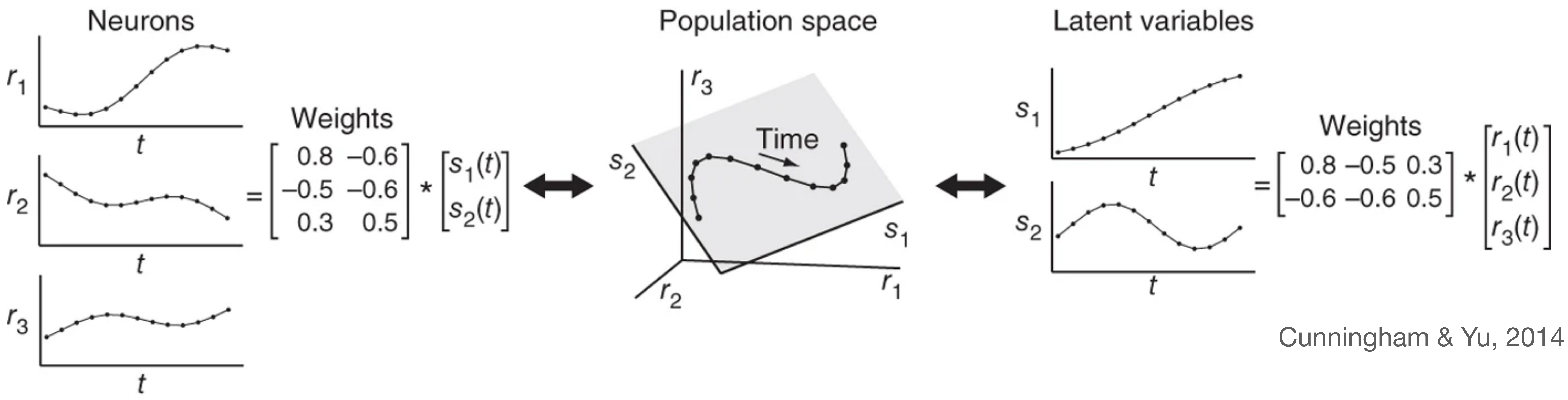
- **Motivation**: Find “latent” (unobserved) structure in neural population activity

- Latents can be discrete or continuous

Overview of Latent Variable Models

- **Motivation**: Find “latent” (unobserved) structure in neural population activity

- Latents can be discrete or **continuous**

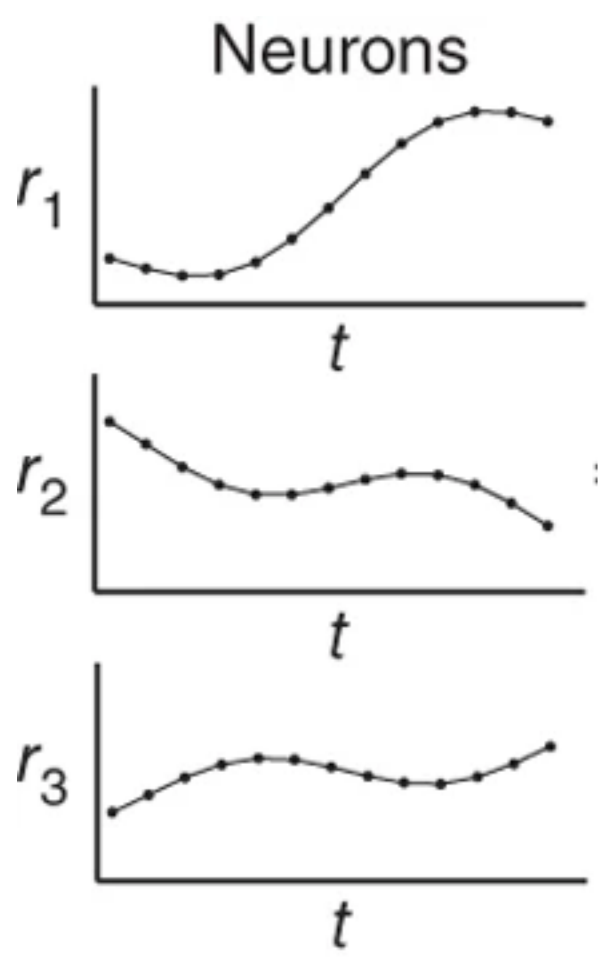


Cunningham & Yu, 2014

Overview of Latent Variable Models

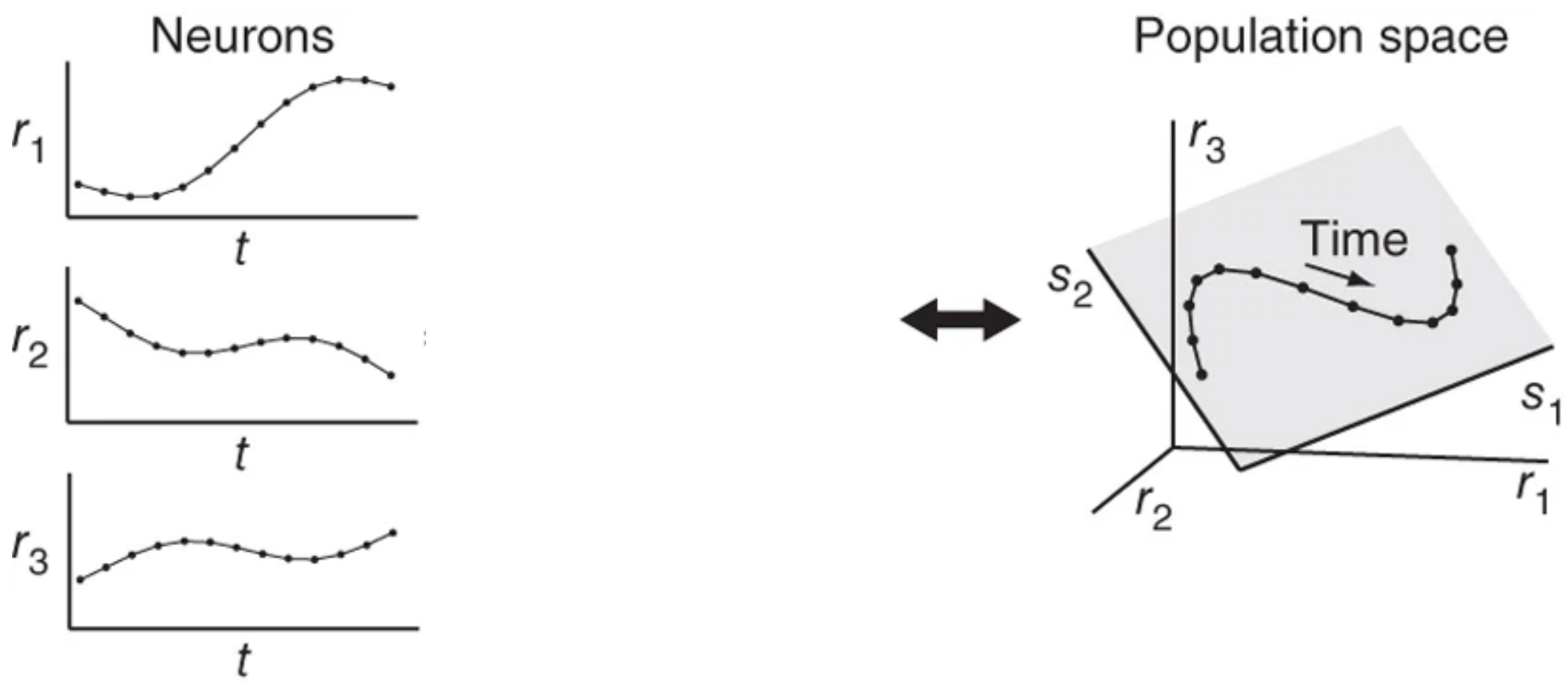
- **Motivation**: Find “latent” (unobserved) structure in neural population activity

- Latents can be discrete or **continuous**



Overview of Latent Variable Models

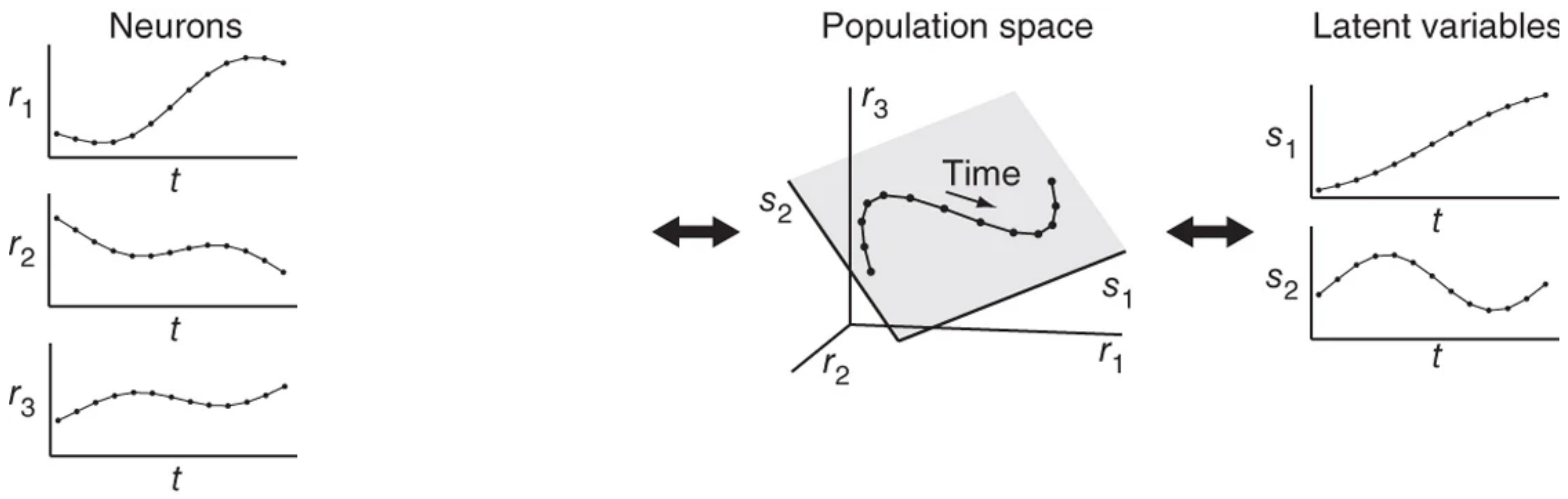
- **Motivation**: Find “latent” (unobserved) structure in neural population activity
- Latents can be discrete or **continuous**



Overview of Latent Variable Models

- **Motivation**: Find “latent” (unobserved) structure in neural population activity

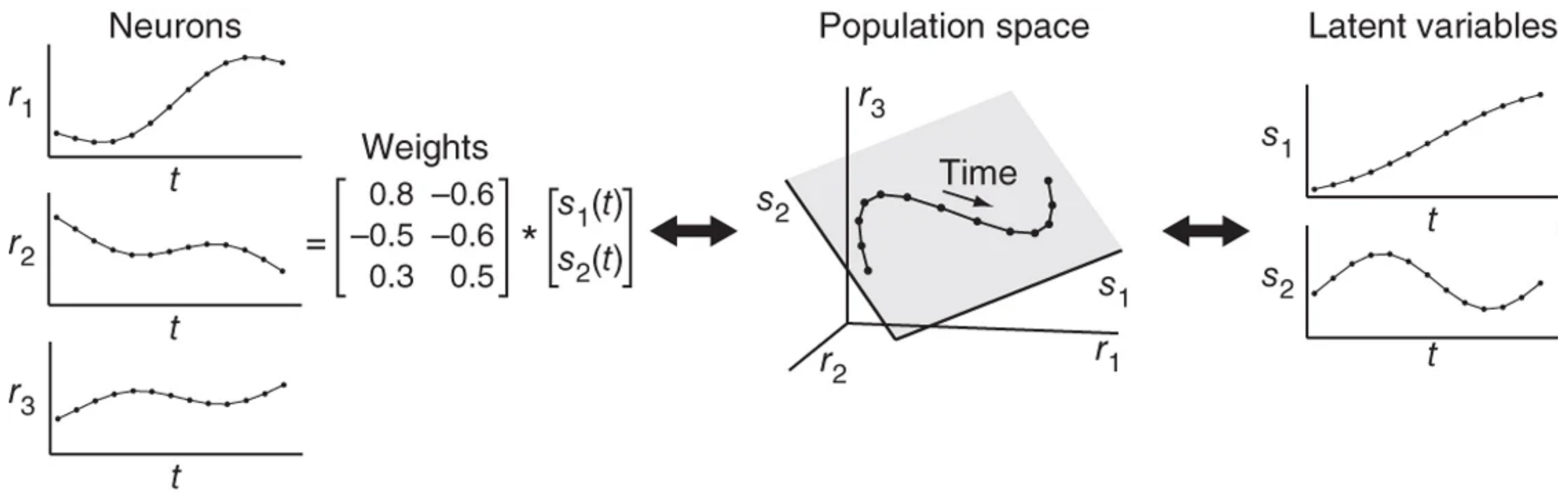
- Latents can be discrete or **continuous**



Overview of Latent Variable Models

- **Motivation**: Find "latent" (unobserved) structure in neural population activity

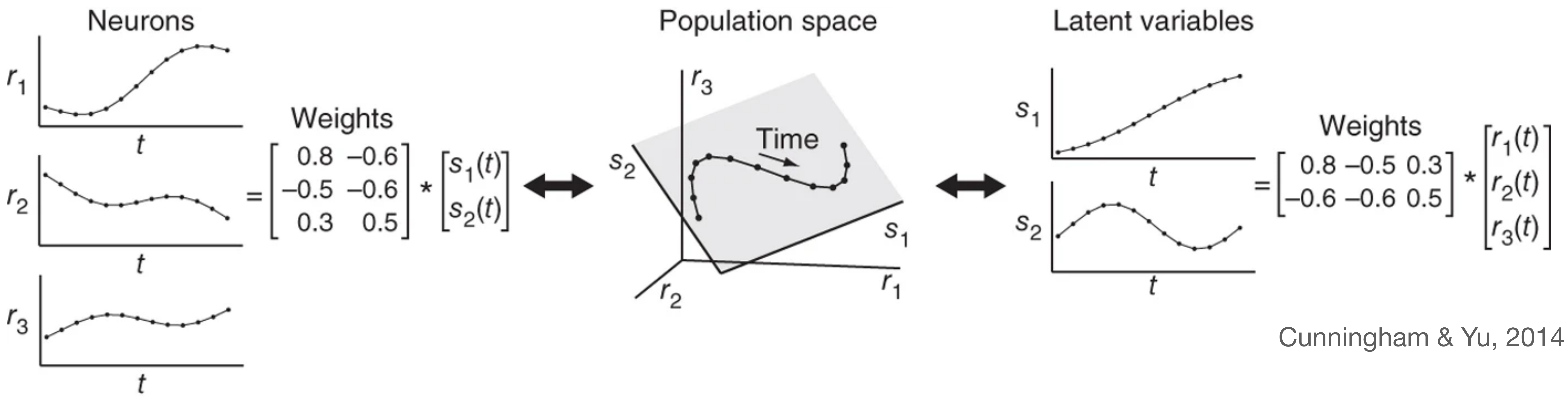
- Latents can be discrete or **continuous**



Overview of Latent Variable Models

- **Motivation**: Find “latent” (unobserved) structure in neural population activity

- Latents can be discrete or **continuous**

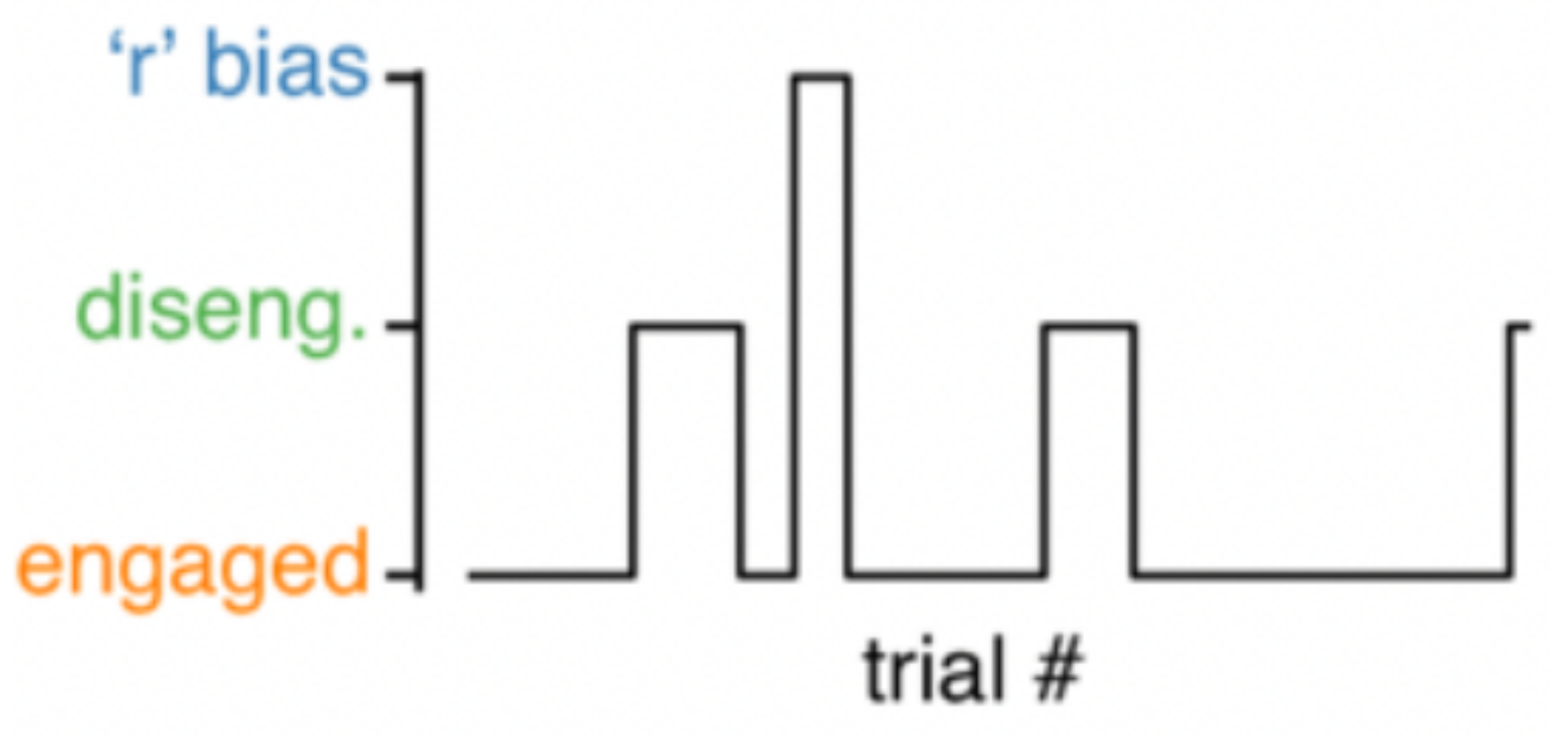


Cunningham & Yu, 2014

Overview of Latent Variable Models

- **Motivation**: Find “latent” (unobserved) structure in neural population activity

- Latents can be **discrete** or continuous

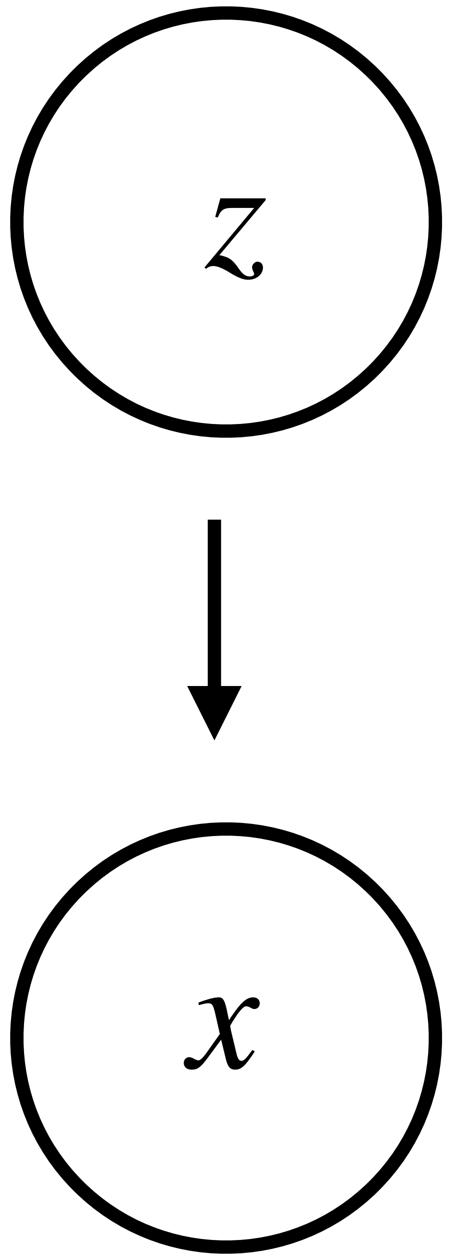


Ashwood et al, 2020

Overview of Workshop

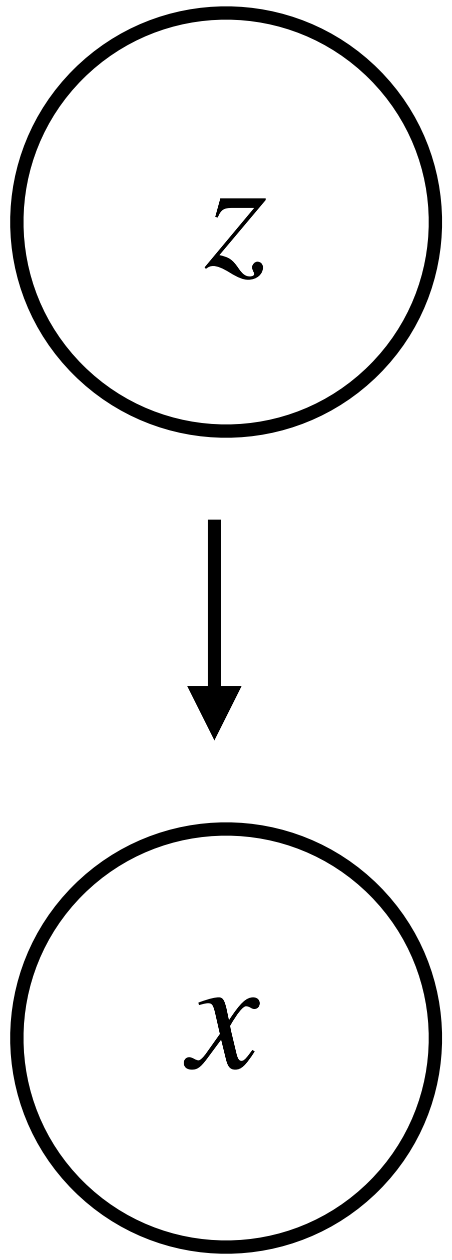
- **Day 1:**
  - Intro to LVMs
  - Factor Analysis (FA)
  - Gaussian Processes (GPs)
  - Gaussian Process Factor Analysis (GPFA)
  - Hidden Markov Models (HMMs)
- **Day 2:**
  - Linear Dynamical Systems (LDSs)
  - Variational Autoencoders (VAEs)
  - Many variants
- **Day 3:**
  - Recap (mini-presentations?) and final questions

Intro to Latent Variable Models



https://pillowlab.princeton.edu/teaching/statneuro2020/notes/notes18_LatentVariableModels.pdf

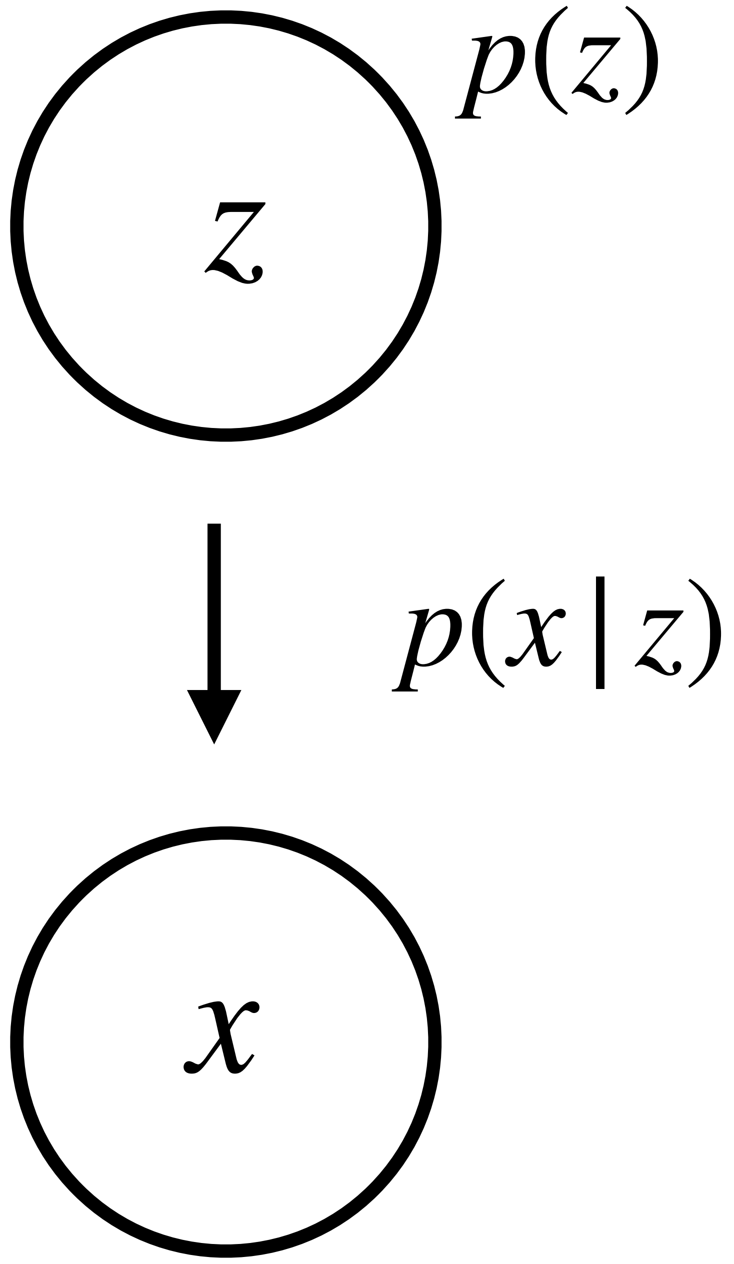
Intro to Latent Variable Models



- Two parts of a LVM
  - Prior: $z \sim p(z)$
  - Conditional probability of observed data: $x \mid z \sim p(x \mid z)$

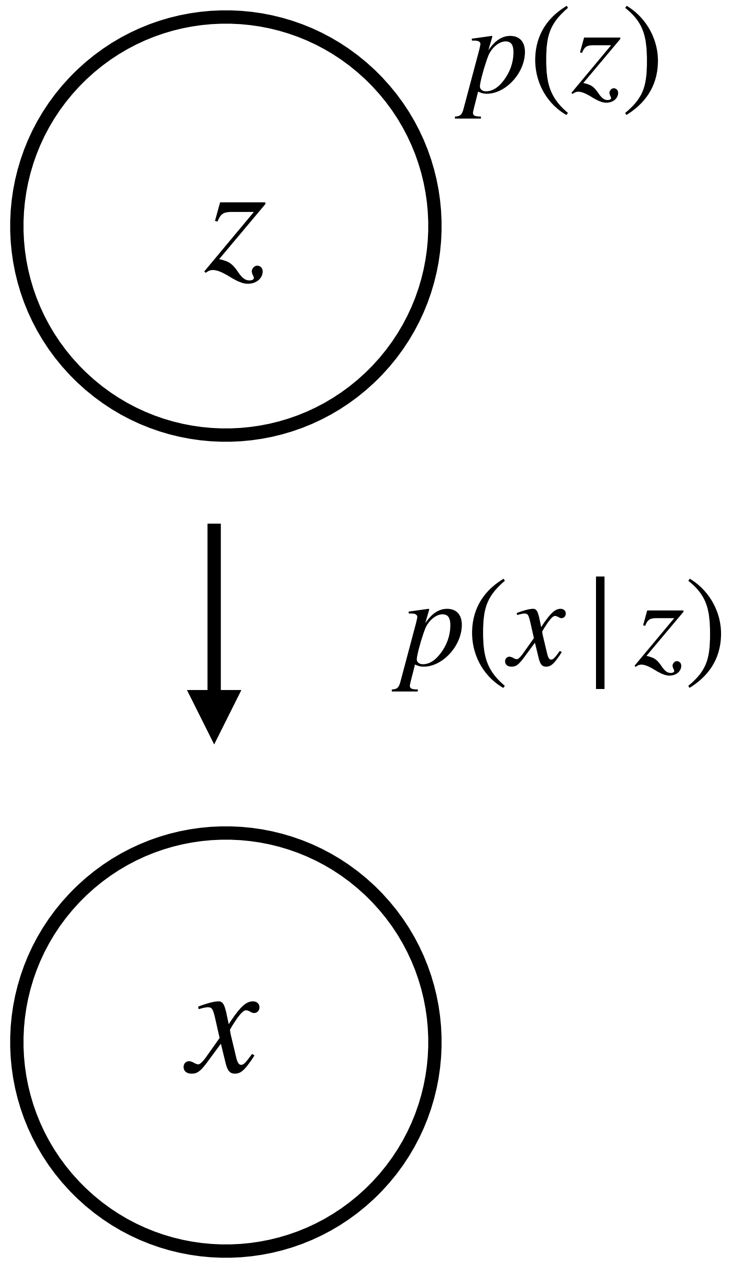
https://pillowlab.princeton.edu/teaching/statneuro2020/notes/notes18_LatentVariableModels.pdf

Intro to Latent Variable Models



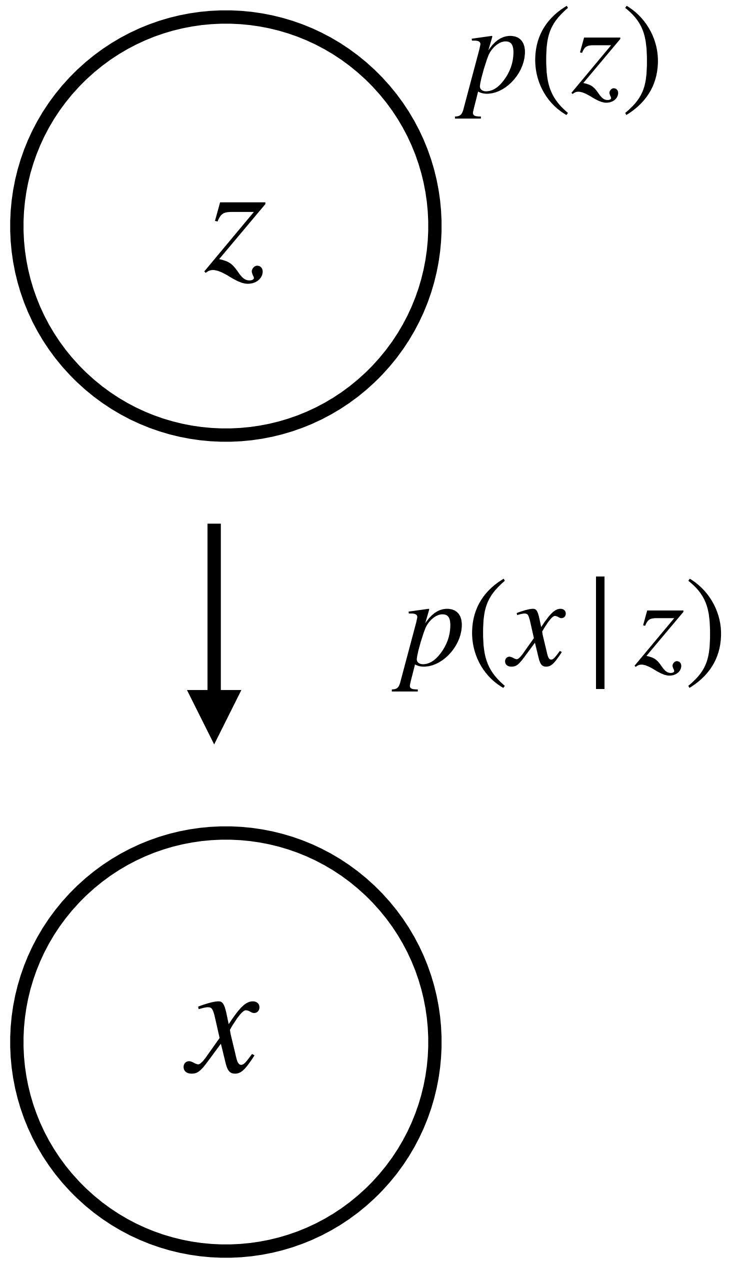
https://pillowlab.princeton.edu/teaching/statneuro2020/notes/notes18_LatentVariableModels.pdf

Intro to Latent Variable Models



- Probability of observed data, $p(x)$, is:

Intro to Latent Variable Models

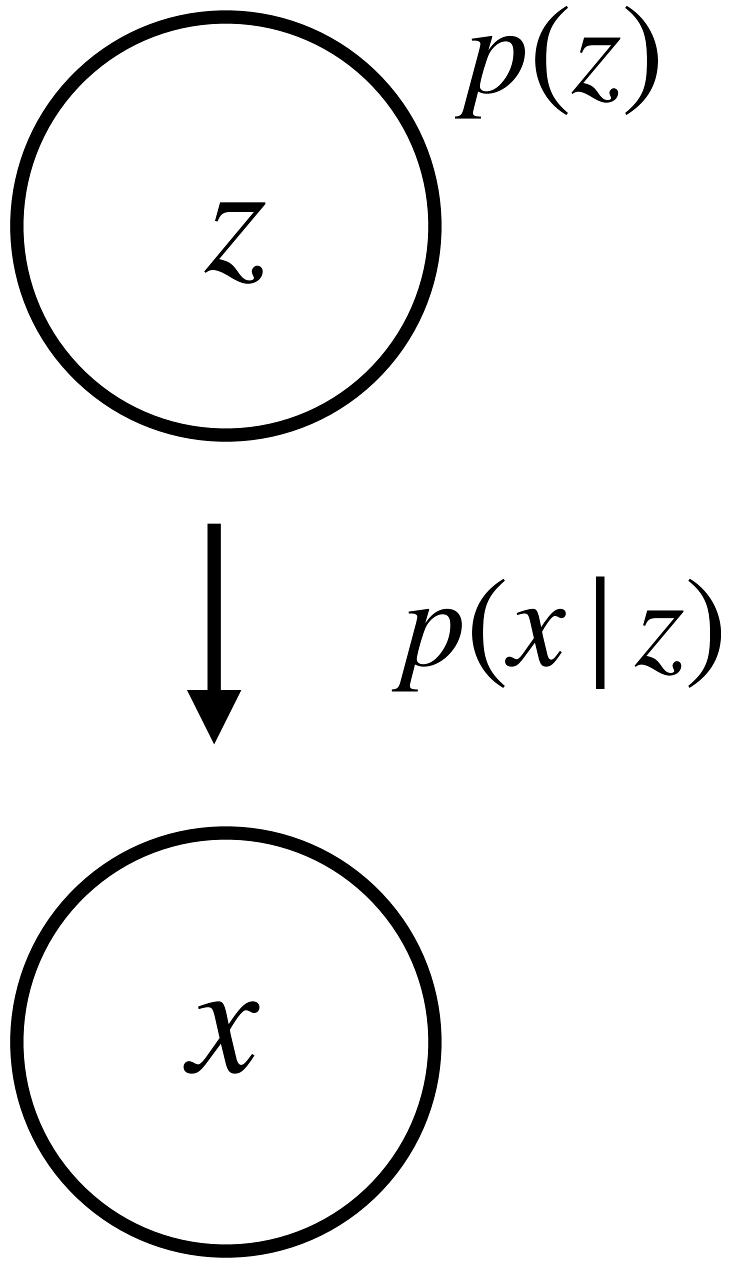


- Probability of observed data, $p(x)$, is:
  - For discrete latents:

$$p(x) = \sum_{i=1}^{m} p(x|z = z_i)p(z = z_i)$$

https://pillowlab.princeton.edu/teaching/statneuro2020/notes/notes18_LatentVariableModels.pdf

Intro to Latent Variable Models



- Probability of observed data, $p(x)$, is:
  - For discrete latents:

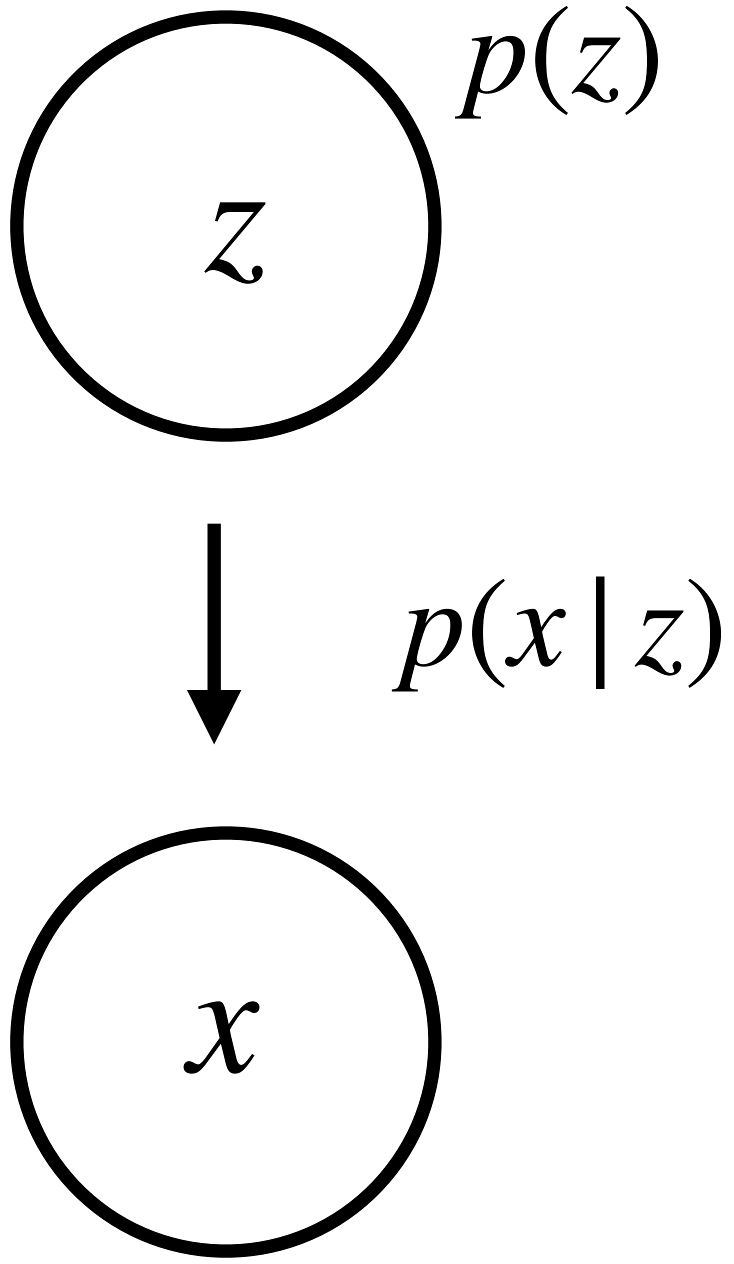
$$p(x) = \sum_{i=1}^{m} p(x|z = z_i)p(z = z_i)$$

  - For continuous latents:

$$p(x) = \int p(x|z)p(z)dz$$

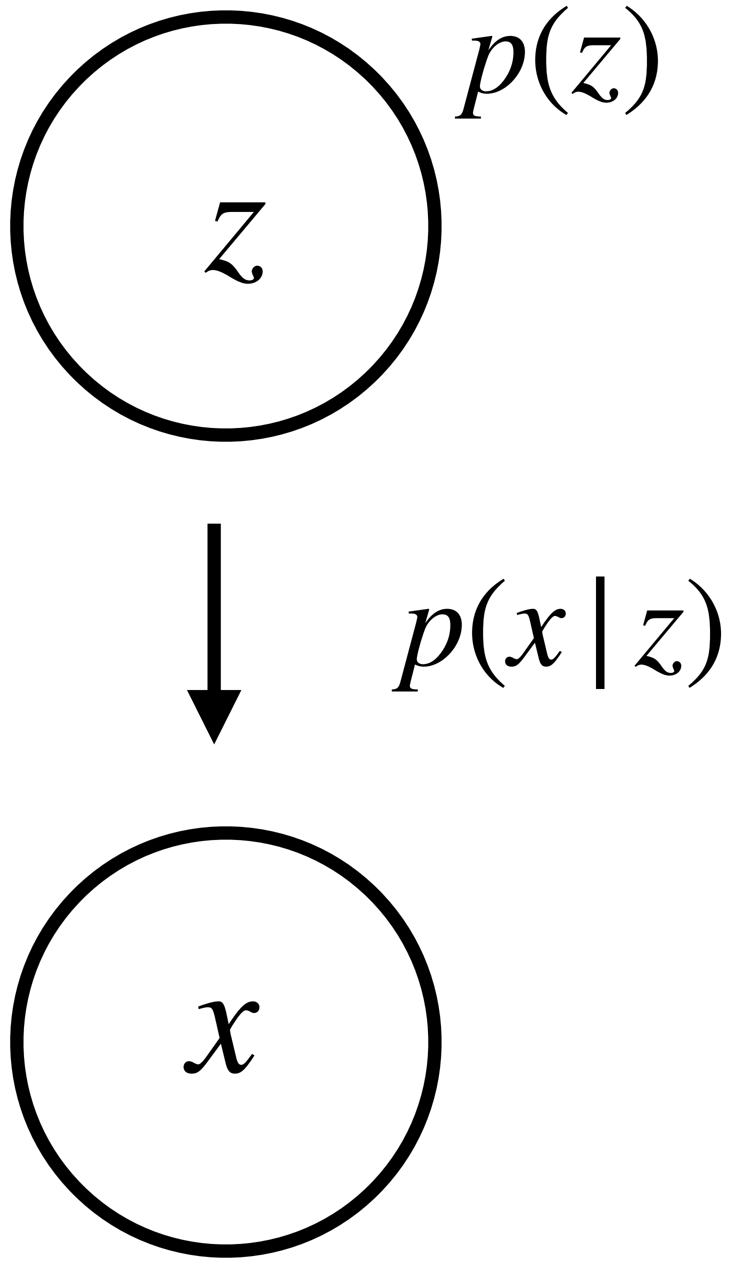
https://pillowlab.princeton.edu/teaching/statneuro2020/notes/notes18_LatentVariableModels.pdf

Intro to Latent Variable Models: Goals



https://pillowlab.princeton.edu/teaching/statneuro2020/notes/notes18_LatentVariableModels.pdf

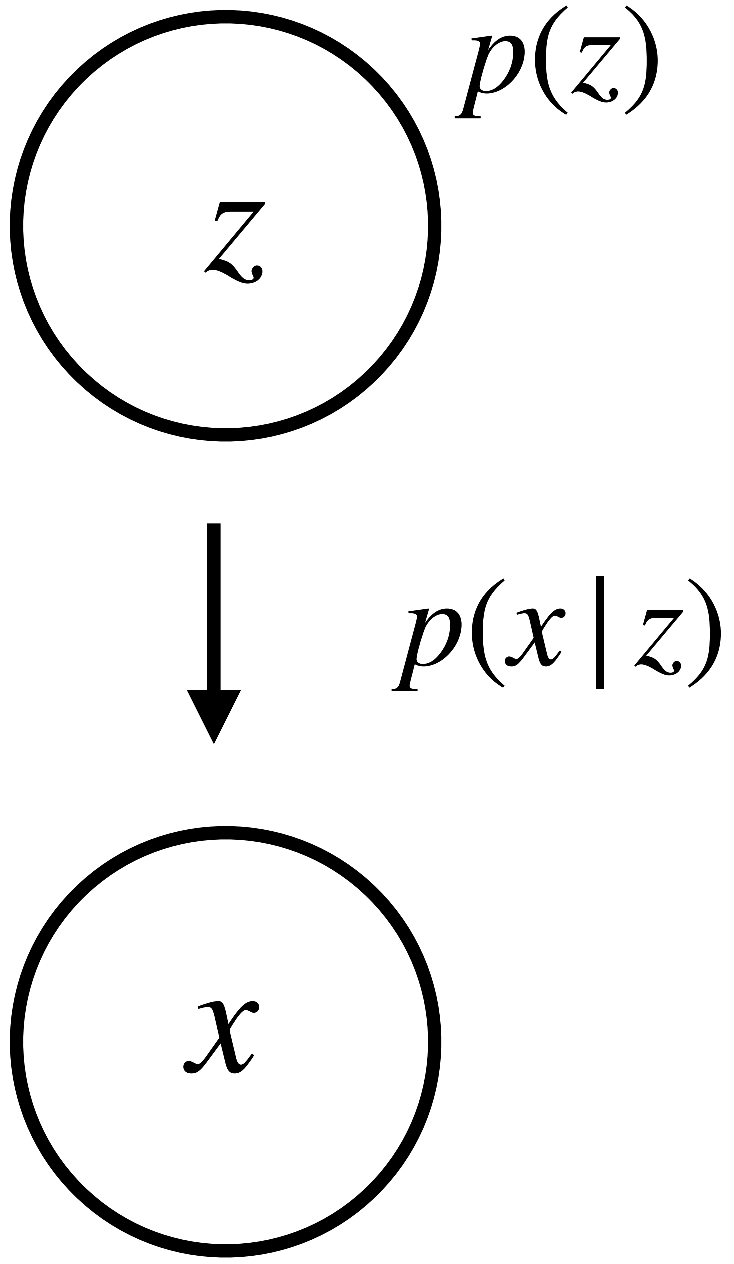
Intro to Latent Variable Models: Goals



- **Recognition/Inference**

$$p(z|x)$$

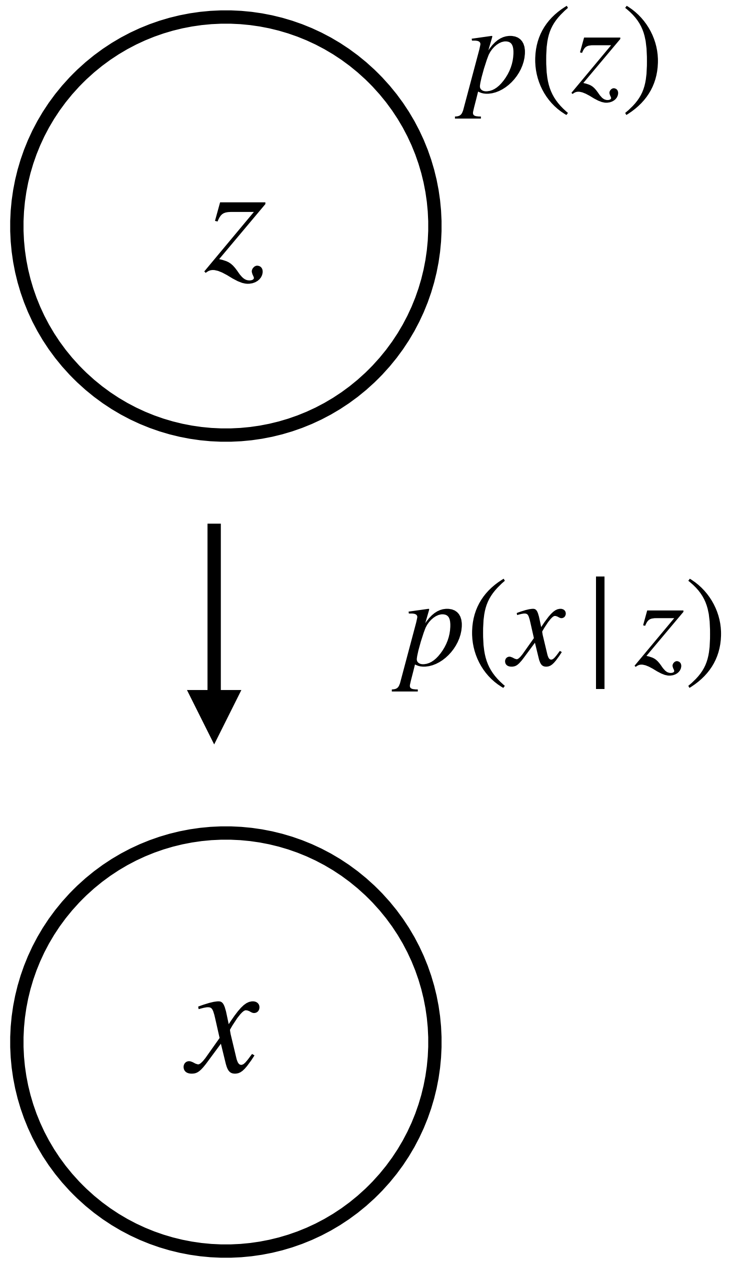
Intro to Latent Variable Models: Goals



- **Recognition/Inference**

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

Intro to Latent Variable Models: Goals



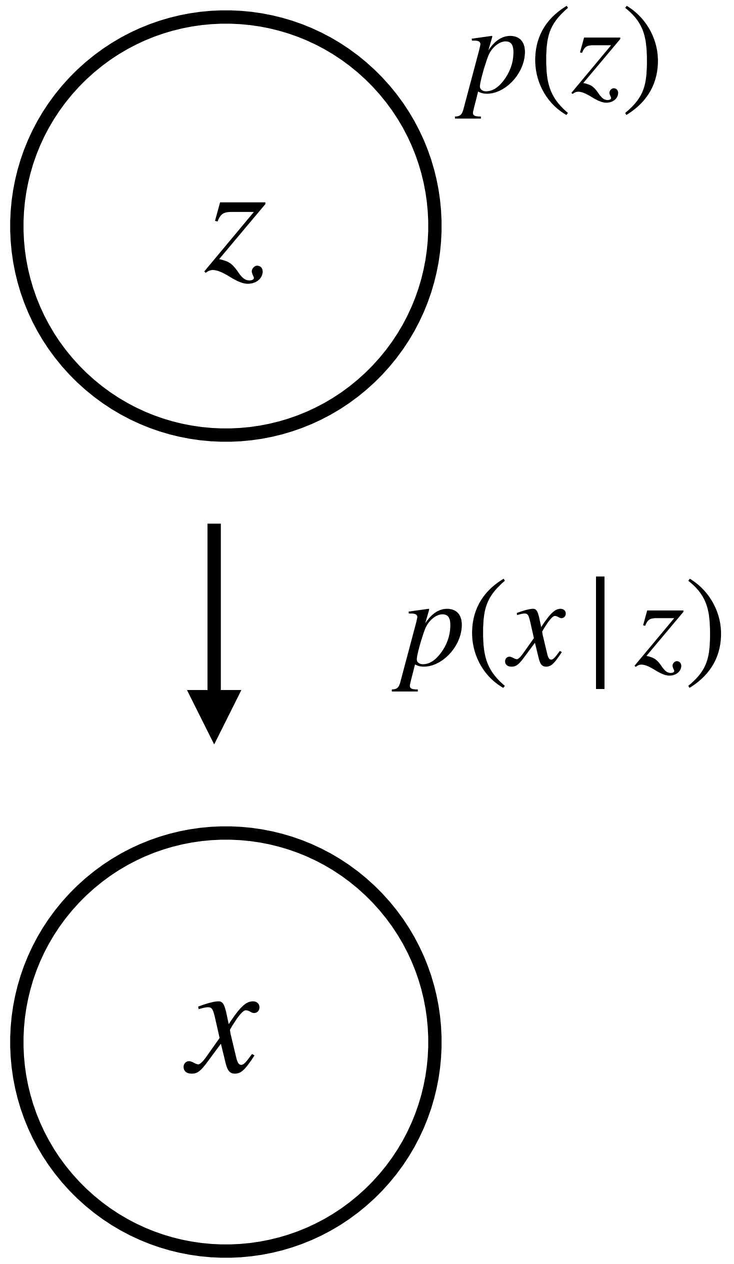
- **Recognition/Inference**

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

- **Model Fitting**
  - Model including parameters is actually:

$$p(x, z|\theta) = p(x|z, \theta)p(z|\theta)$$

Intro to Latent Variable Models: Goals



- **Recognition/Inference**

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

- **Model Fitting**
  - Model including parameters is actually:

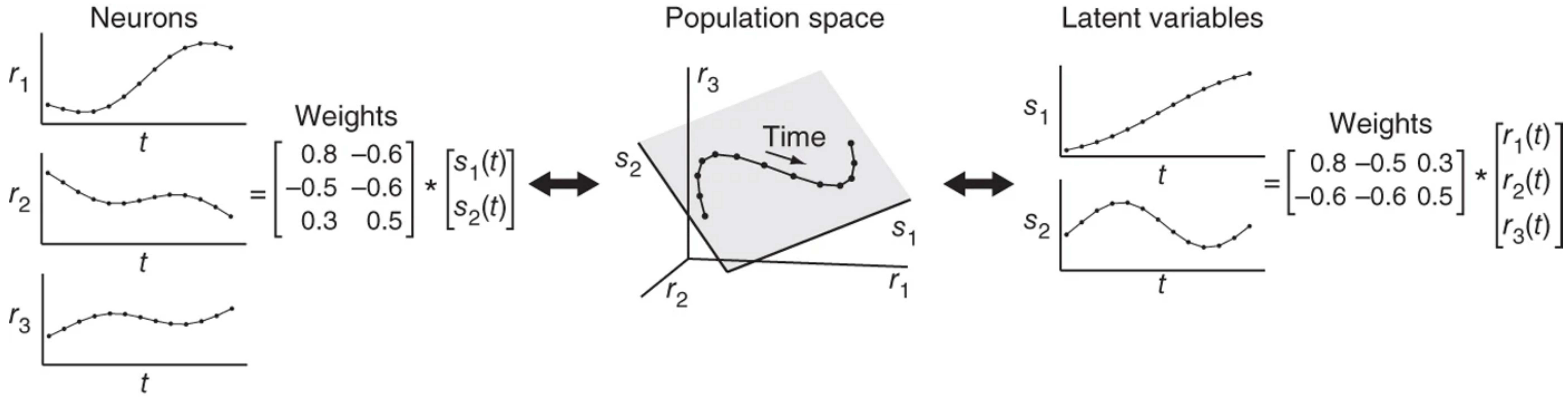
$$p(x, z|\theta) = p(x|z, \theta)p(z|\theta)$$

  - Learning parameters by maximum likelihood:

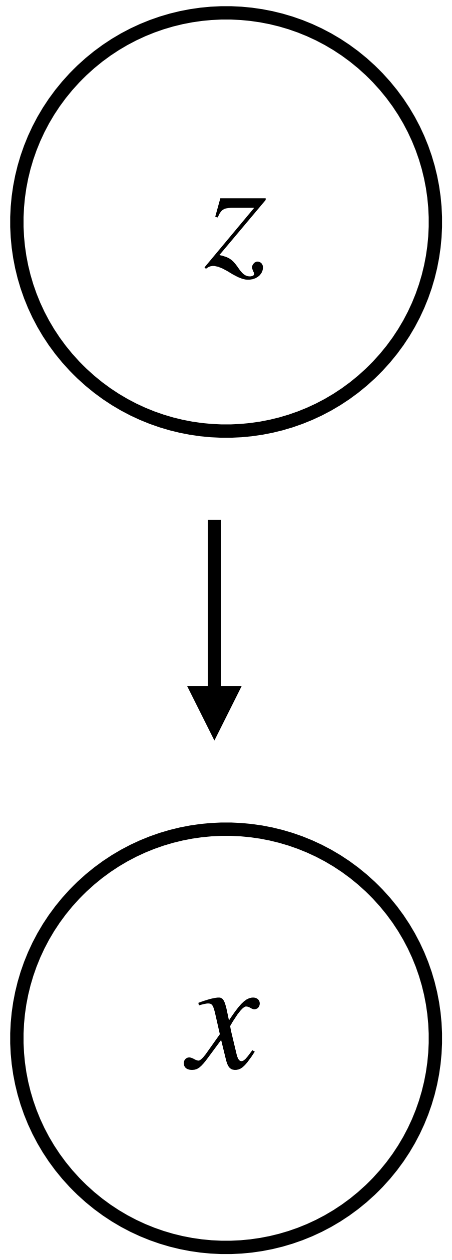
$$\hat{\theta} = \arg\max_{\theta} p(x|\theta) = \arg\max_{\theta} \int p(x, z|\theta)dz.$$

Factor Analysis

Factor Analysis



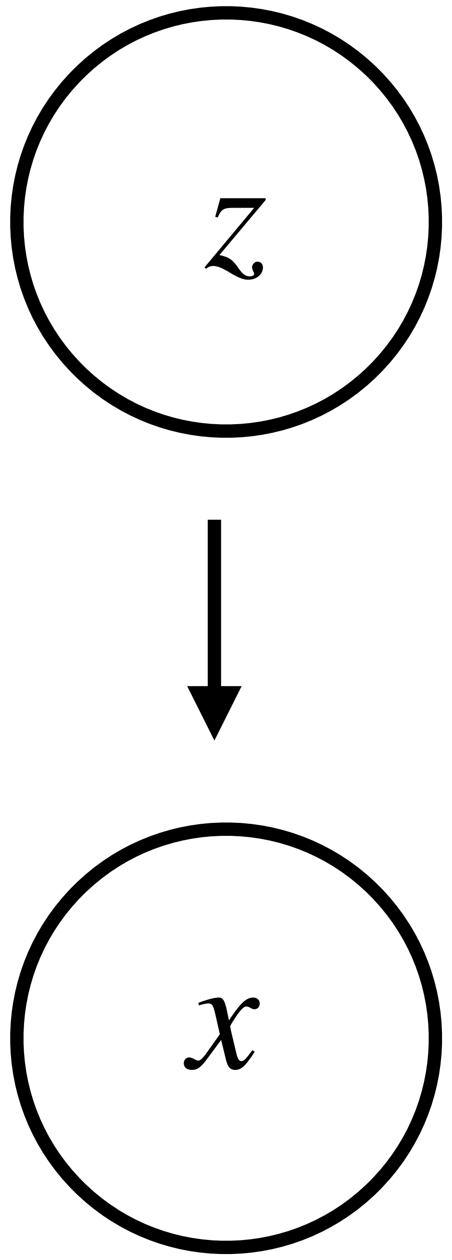
Factor Analysis: Generative Model



$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$$

- Can be any Gaussian (see Murphy, book 1, section 20.2)

Factor Analysis: Generative Model



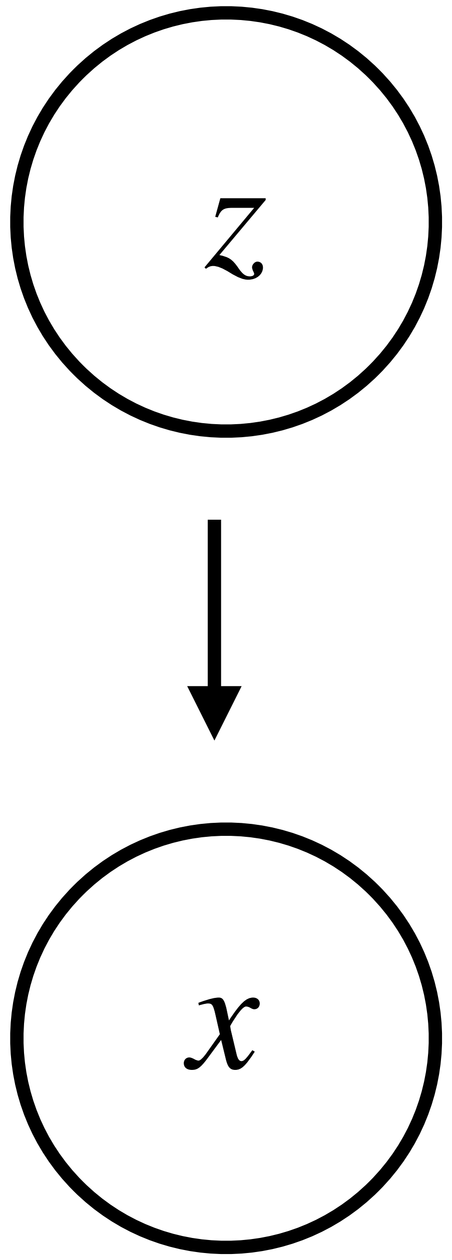
$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi})$$

- Can be any Gaussian (see Murphy, book 1, section 20.2)

- Linear Gaussian model
- $z: \quad D \times T$ latent. dim x samples (timepoints)
- $x: \quad N \times T$ obs. dim (neurons) x samples (timepoints)
- $\mathbf{W}: \quad N \times D$ obs. dim. (neurons) x latent dim.
- $\mathbf{\Psi}: \quad N \times N$ diagonal covariance matrix

Factor Analysis: Generative Model



$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$$

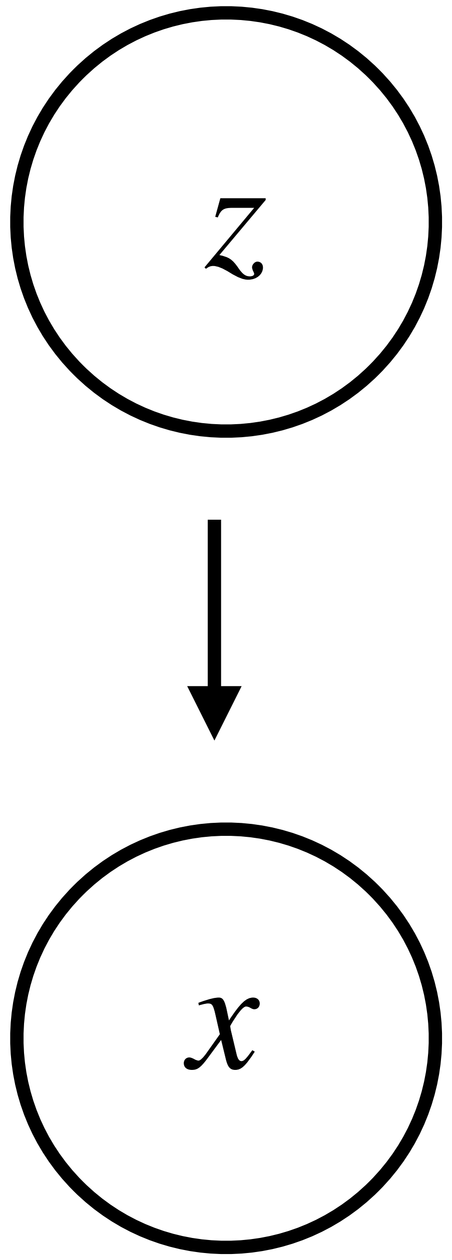
$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi})$$

$$p(x) = \int p(x|z)p(z)dz$$

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^\mathsf{T} + \boldsymbol{\Psi})$$

- Can be any Gaussian (see Murphy, book 1, section 20.2)
- Linear Gaussian model
- z : $D \times T$ latent. dim x samples (timepoints)
- x : $N \times T$ obs. dim (neurons) x samples (timepoints
- $\mathbf{W}$: $N \times D$ obs. dim. (neurons) x latent dim.
- $\boldsymbol{\Psi}$: $N \times N$ diagonal covariance matrix

Factor Analysis: Generative Model



$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi})$$

$$p(x) = \int p(x|z)p(z)dz$$

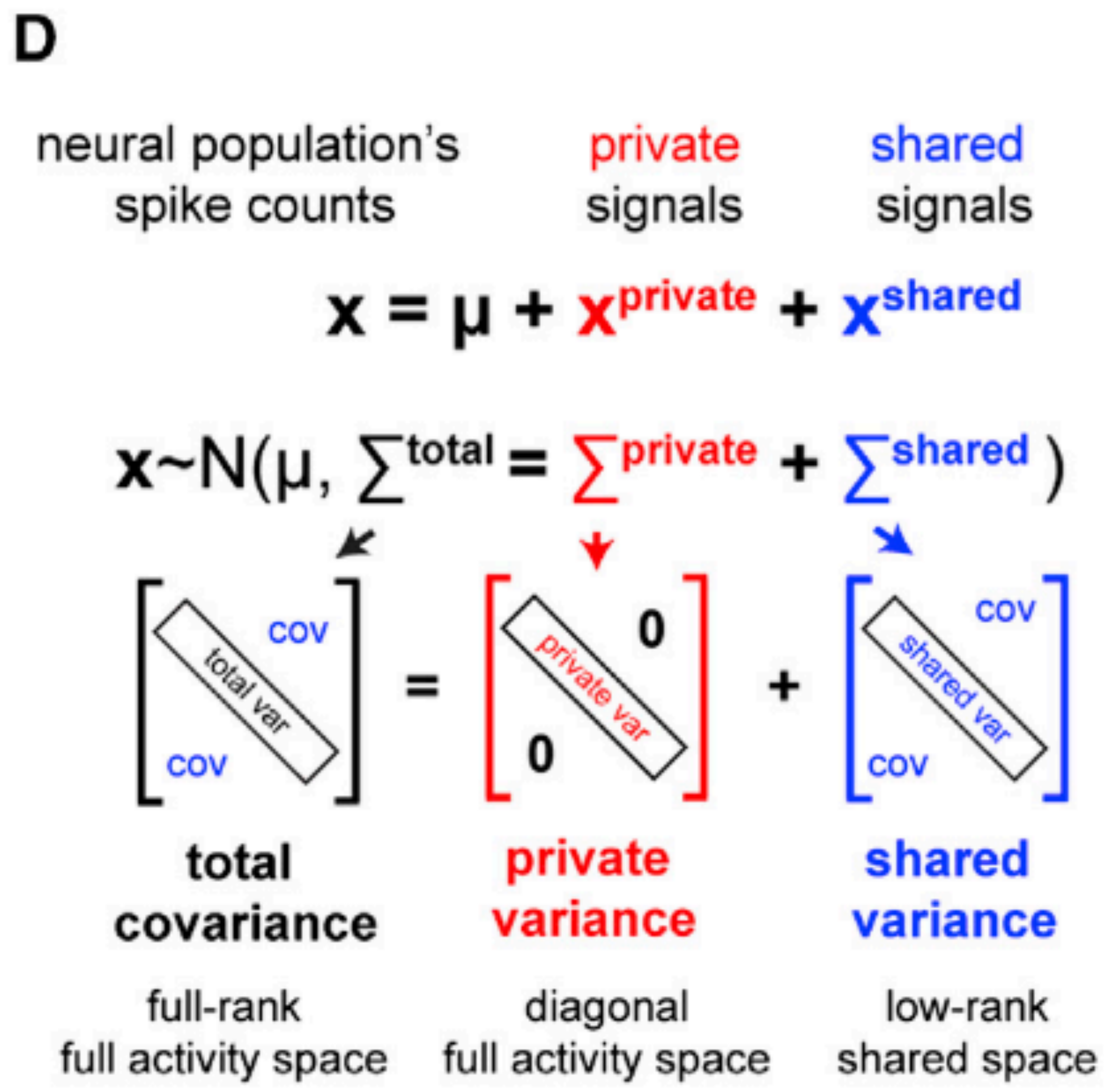
$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^{\mathsf{T}} + \boldsymbol{\Psi})$$

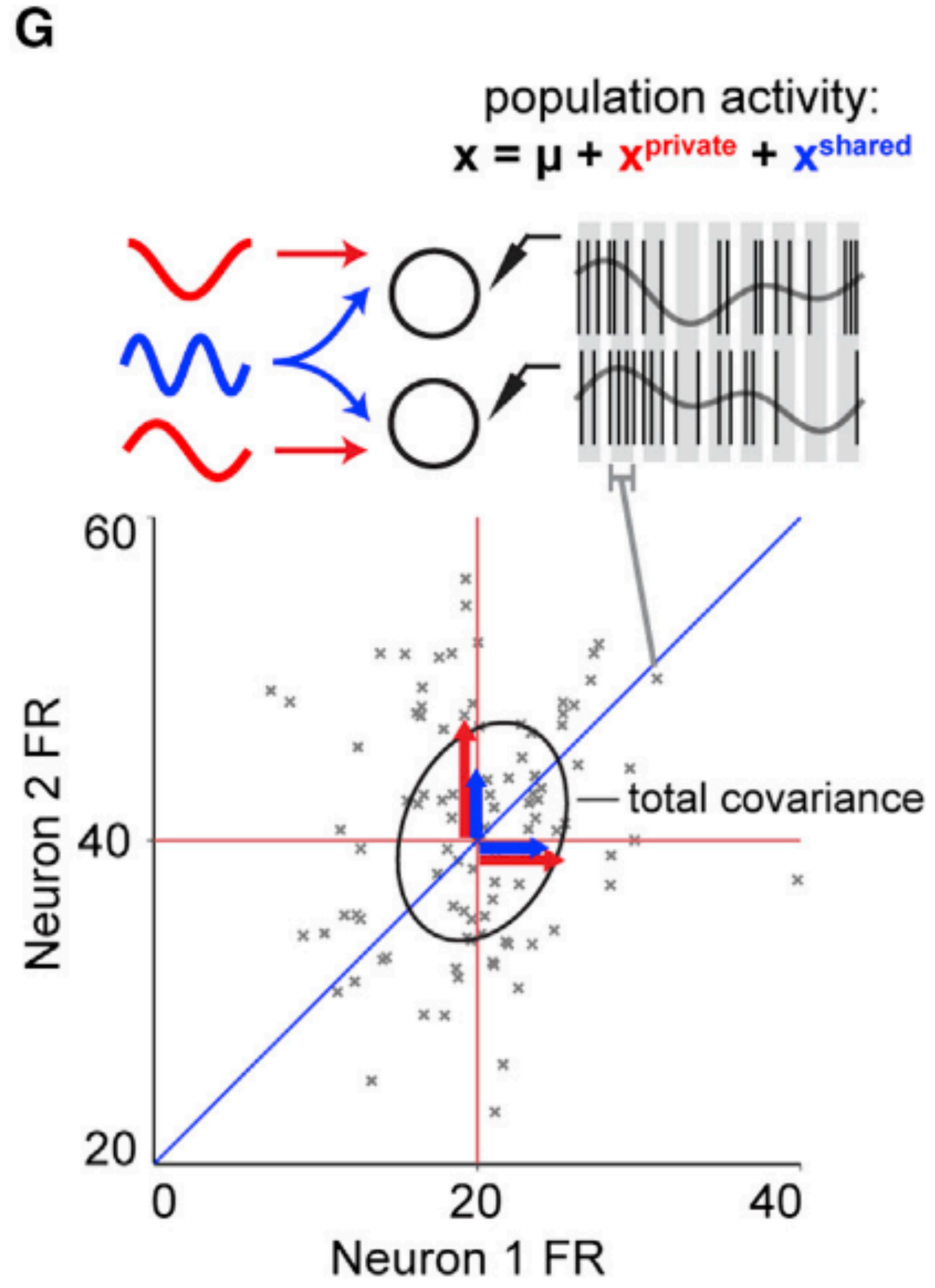
Low Rank + Noise

Shared and Unique

- Can be any Gaussian (see Murphy, book 1, section 20.2)
- Linear Gaussian model
- z : $D \times T$ latent. dim x samples (timepoints)
- x : $N \times T$ obs. dim (neurons) x samples (timepoints
- $\mathbf{W}$: $N \times D$ obs. dim. (neurons) x latent dim.
- $\boldsymbol{\Psi}$: $D \times D$ diagonal covariance matrix

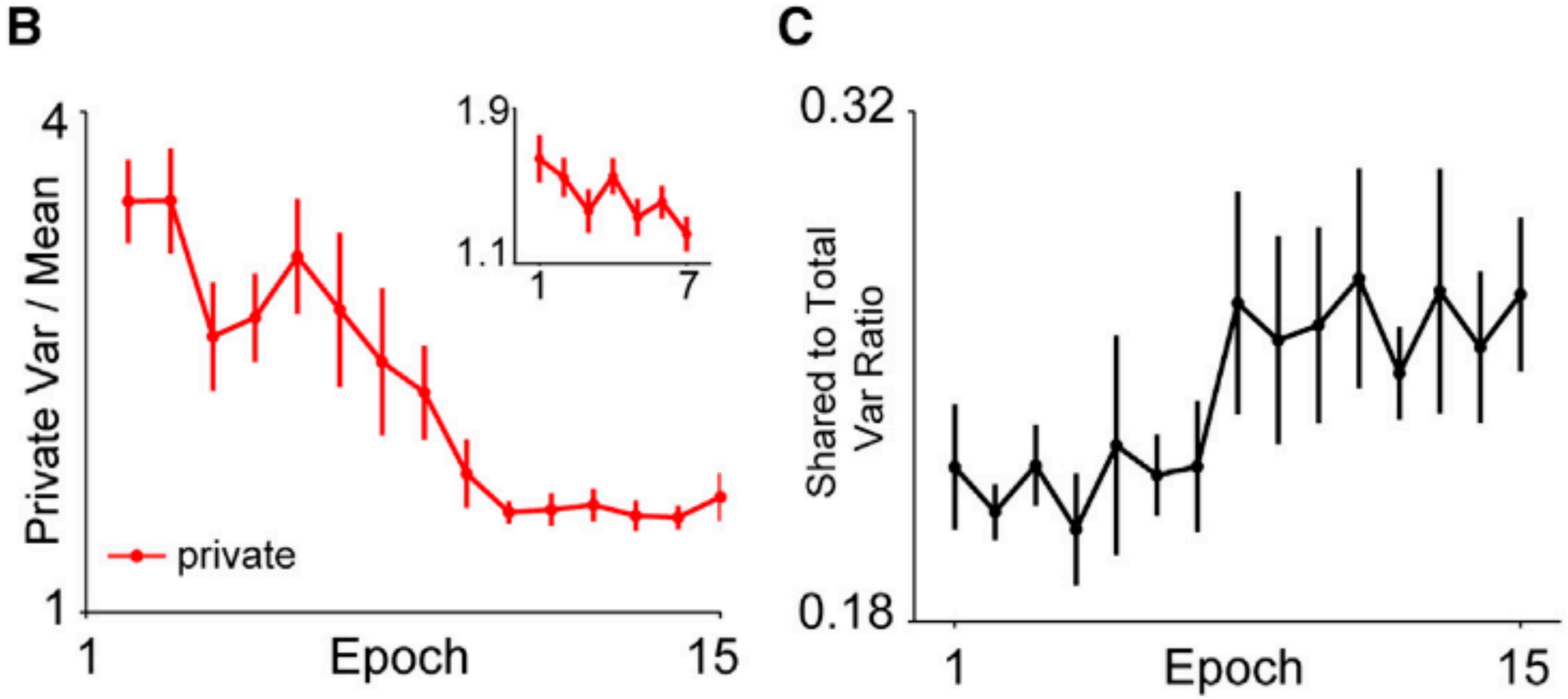
Factor Analysis: Shared and Unique Example





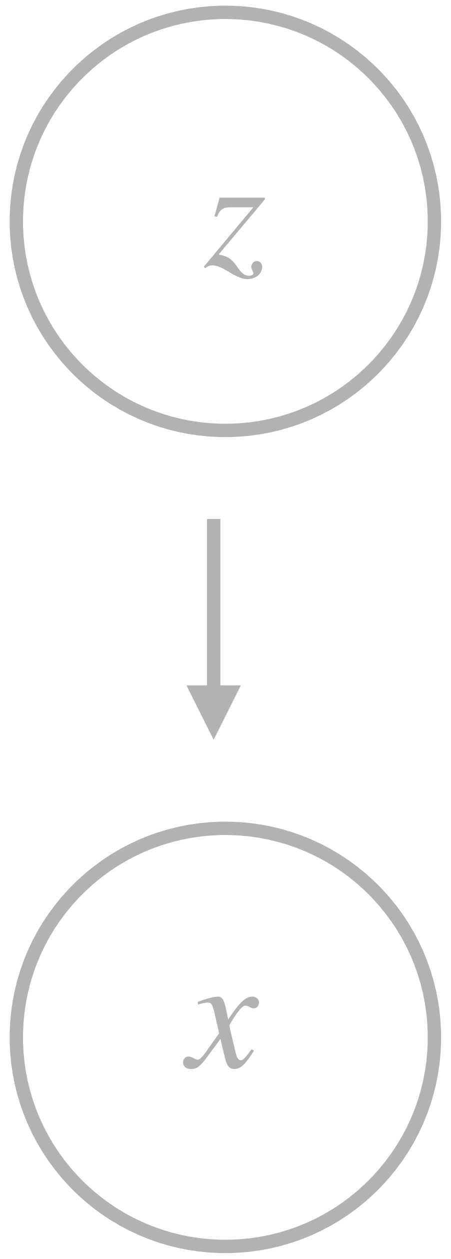
Athalye et al, Neuron 2017

Factor Analysis: Shared and Unique Example



Athalye et al, Neuron 2017

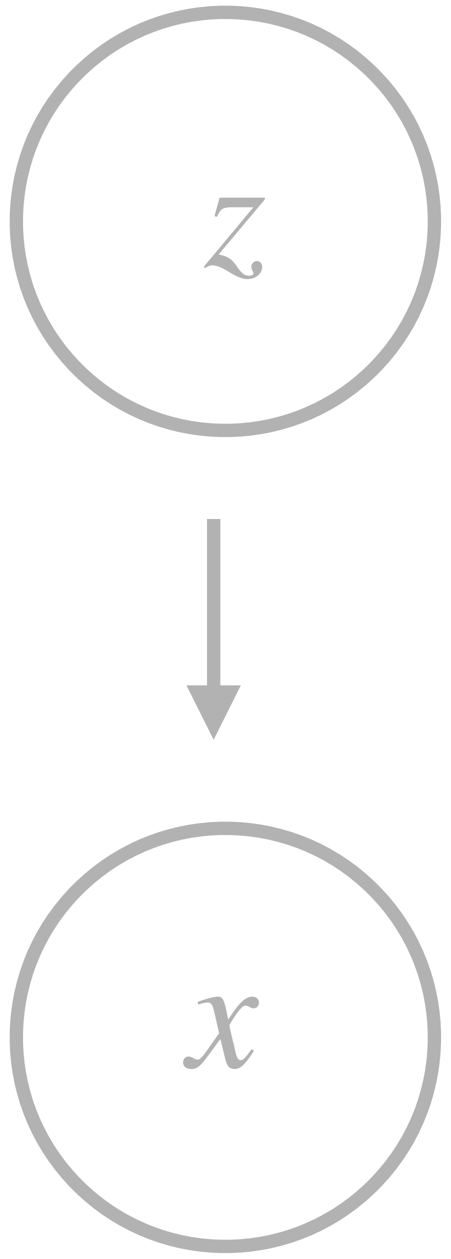
FA vs. Probabilistic PCA vs. PCA



$$p(z) = \mathcal{N}(z|\mathbf{0}, \mathbf{I})$$

$$p(x|z) = \mathcal{N}(x|\mathbf{W}z + \boldsymbol{\mu}, \boldsymbol{\Psi})$$

FA vs. Probabilistic PCA vs. PCA

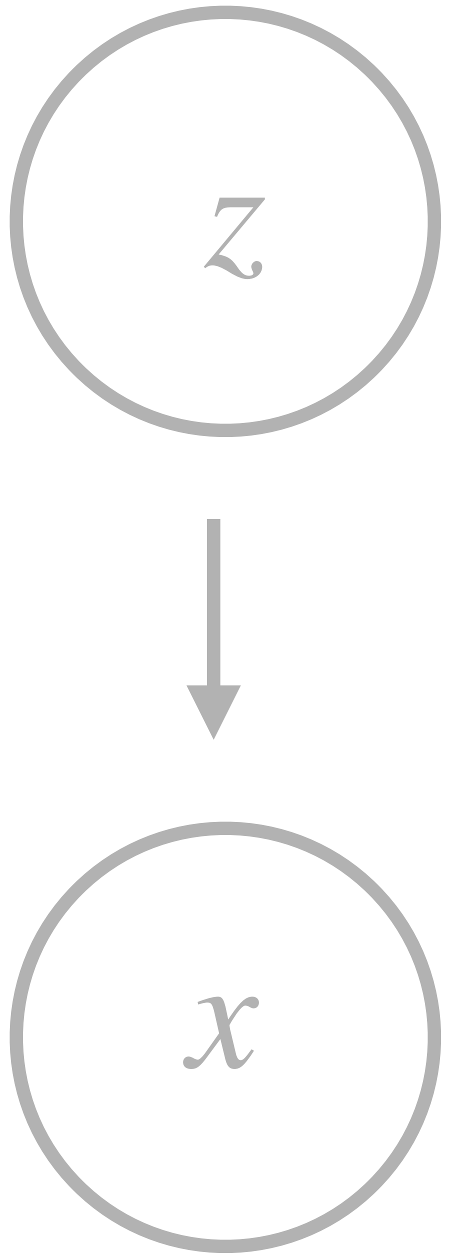


$$p(z) = \mathcal{N}(z|\mathbf{0}, \mathbf{I})$$

$$p(x|z) = \mathcal{N}(x|\mathbf{W}z + \mu, \Psi)$$

- Probabilistic PCA is Factor Analysis where $\boldsymbol{\Psi}$ is the identity matrix (all observations have the same independent noise)

FA vs. Probabilistic PCA vs. PCA

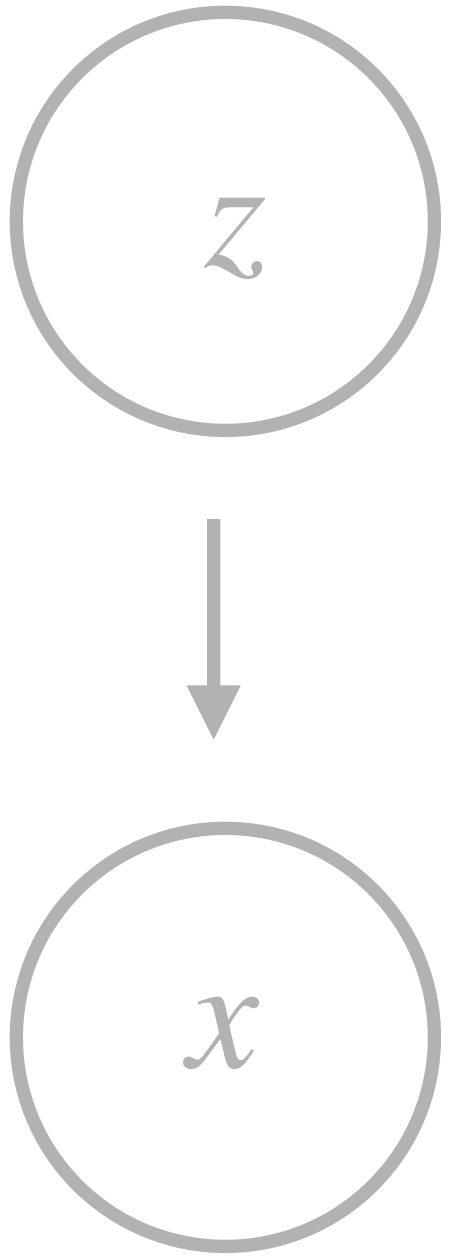


$$p(z) = \mathcal{N}(z|\mathbf{0}, \mathbf{I})$$

$$p(x|z) = \mathcal{N}(x|\mathbf{W}z + \mu, \Psi)$$

- Probabilistic PCA is Factor Analysis where $\boldsymbol{\Psi}$ is the identity matrix (all observations have the same independent noise)
- PPCA when $\boldsymbol{\Psi}$ -> 0 becomes PCA

FA vs. PCA: An Example

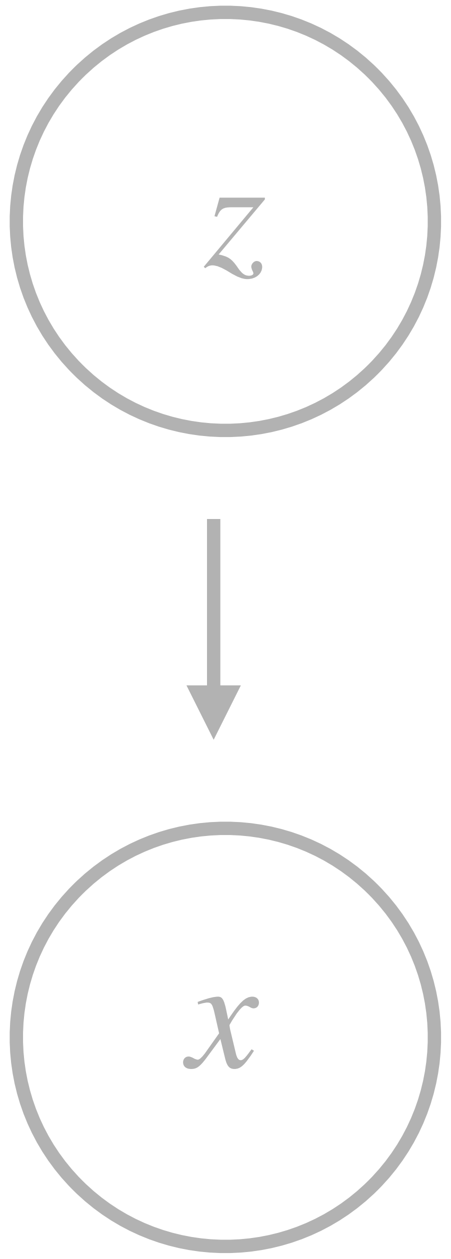


$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi})$$

$$W = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \Psi = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$$

FA vs. PCA: An Example



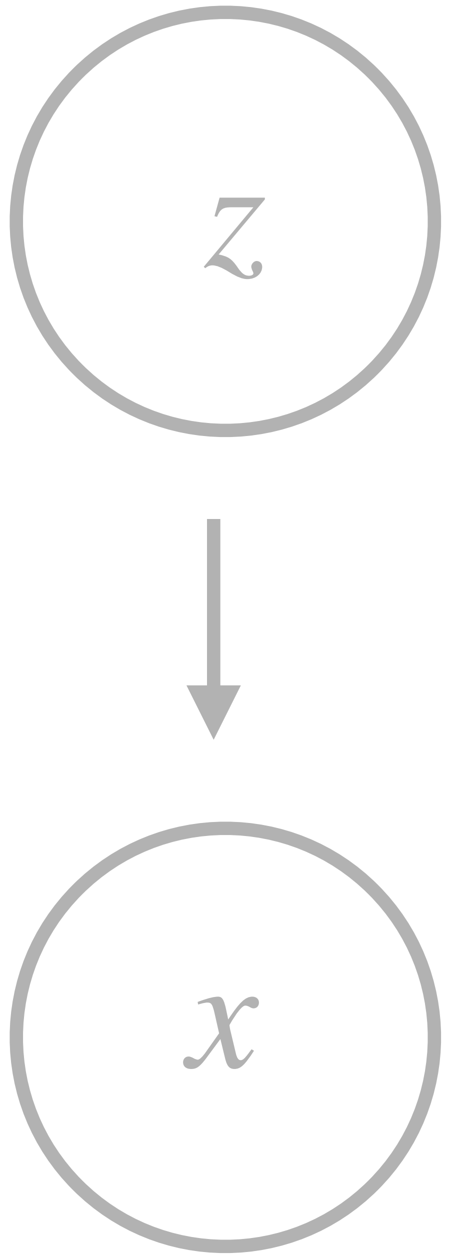
$p(z) = \mathcal{N}(z|\mathbf{0}, \mathbf{I})$

$p(x|z) = \mathcal{N}(x|\mathbf{W}z + \mu, \Psi)$

$$W = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \Psi = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\text{cov}(X) = WW^T + \Psi = \begin{bmatrix} 101 & 1 \\ 1 & 2 \end{bmatrix}$$

FA vs. PCA: An Example



$$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}|\mathbf{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Psi})$$

$$W = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \Psi = \begin{bmatrix} 100 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\text{cov}(X) = WW^T + \Psi = \begin{bmatrix} 101 & 1 \\ 1 & 2 \end{bmatrix}$$

- PCA would give a top eigenvector primarily lying along the first dimension

Factor Analysis: Inferring the latents

$$p(z \mid x) \propto p(x \mid z)p(z)$$

$$= \mathcal{N}(x \mid Wz, \Psi) \cdot \mathcal{N}(z \mid 0, I)$$

Factor Analysis: Inferring the latents

$$p(z \mid x) \propto p(x \mid z) p(z)$$

$$= \mathcal{N}(x \mid Wz, \Psi) \cdot \mathcal{N}(z \mid 0, I)$$

$$\vdots$$

$$= \mathcal{N}(\Lambda W^T \Psi^{-1} x, \Lambda)$$ where $$\Lambda = \left(W^T \Psi^{-1} W + I\right)^{-1}$$

Factor Analysis: Inferring the latents

$$p(z \mid x) \propto p(x \mid z)p(z)$$

$$= \mathcal{N}(x \mid Wz, \Psi) \cdot \mathcal{N}(z \mid 0, I)$$

$$\vdots$$

$$= \mathcal{N}(\Lambda W^T \Psi^{-1} x, \Lambda) \quad \text{where} \quad \Lambda = \left(W^T \Psi^{-1} W + I\right)^{-1}$$

- When inferring the latent, the components of x are downweighted in proportion to their amount of independent noise (value in Ψ).

EM for Factor Analysis

- E step: Estimate the posterior, $p(z|x)$, given set parameters
- M step: Estimate the parameters, $[W, \Psi]$, given the expectations of the latents

Why probabilistic models, versus PCA?

- Allows having more sophisticated, and more accurate models
    - Different noise models (FA vs PPCA), mixture of factor analyzers, etc…
- Principled
- Better for missing data, or streaming data
- FA won’t be as dependent on scaling

- However, PCA has simpler understanding in terms of variance and orthogonality, and is faster to run!
- Important to check that scientific results are robust across methods

Gaussian Processes

Gaussian Processes

Now consider a function $f : \mathcal{X} \to \mathbb{R}$ evaluated at a set of inputs, $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{X}\}_{n=1}^{N}$. Let $\boldsymbol{f}_X = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]$ be the set of unknown function values at these points.

Gaussian Processes

Now consider a function $f : \mathcal{X} \to \mathbb{R}$ evaluated at a set of inputs, $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{X}\}_{n=1}^{N}$. Let $\boldsymbol{f}_X = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]$ be the set of unknown function values at these points. If $\boldsymbol{f}_X$ is jointly Gaussian for any set of $N \geq 1$ points, then we say that $f : \mathcal{X} \to \mathbb{R}$ is a **Gaussian process**.

Gaussian Processes

Now consider a function $f : \mathcal{X} \to \mathbb{R}$ evaluated at a set of inputs, $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{X}\}_{n=1}^N$. Let $\boldsymbol{f}_X = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]$ be the set of unknown function values at these points. If $\boldsymbol{f}_X$ is jointly Gaussian for any set of $N \geq 1$ points, then we say that $f : \mathcal{X} \to \mathbb{R}$ is a **Gaussian process**. Such a process is defined by its **mean function** $m(\boldsymbol{x}) \in \mathbb{R}$ and a **covariance function**, $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \geq 0$, which is any positive definite **Mercer kernel**

Gaussian Processes

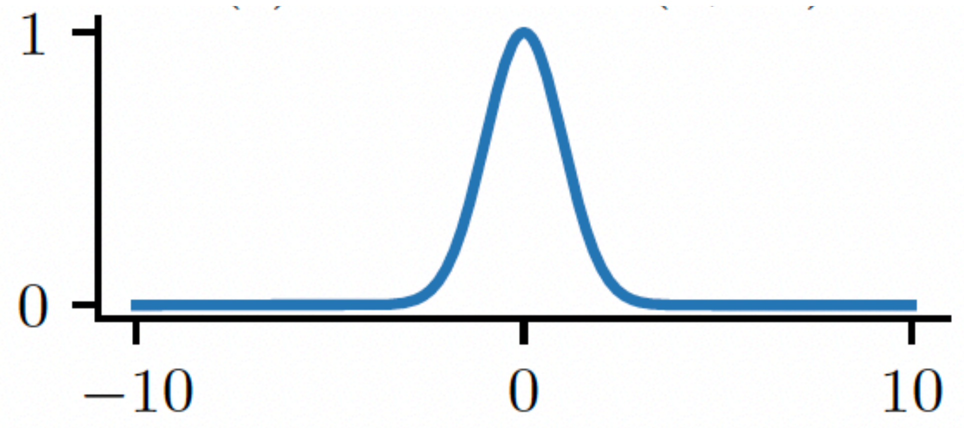
Now consider a function $f : \mathcal{X} \rightarrow \mathbb{R}$ evaluated at a set of inputs, $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{X}\}_{n=1}^{N}$. Let $\boldsymbol{f}_X = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]$ be the set of unknown function values at these points. If $\boldsymbol{f}_X$ is jointly Gaussian for any set of $N \geq 1$ points, then we say that $f : \mathcal{X} \rightarrow \mathbb{R}$ is a **Gaussian process**. Such a process is defined by its **mean function** $m(\boldsymbol{x}) \in \mathbb{R}$ and a **covariance function**, $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \geq 0$, which is any positive definite **Mercer kernel**

- Example Kernel ("Radial Basis Function"): $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'; \ell) = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\ell^2}\right)$

Gaussian Processes

Now consider a function $f : \mathcal{X} \to \mathbb{R}$ evaluated at a set of inputs, $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{X}\}_{n=1}^{N}$. Let $\boldsymbol{f}_X = [f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_N)]$ be the set of unknown function values at these points. If $\boldsymbol{f}_X$ is jointly Gaussian for any set of $N \geq 1$ points, then we say that $f : \mathcal{X} \to \mathbb{R}$ is a **Gaussian process**. Such a process is defined by its **mean function** $m(\boldsymbol{x}) \in \mathbb{R}$ and a **covariance function**, $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \geq 0$, which is any positive definite **Mercer kernel**

- Example Kernel ("Radial Basis Function"): $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'; \ell) = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\ell^2}\right)$

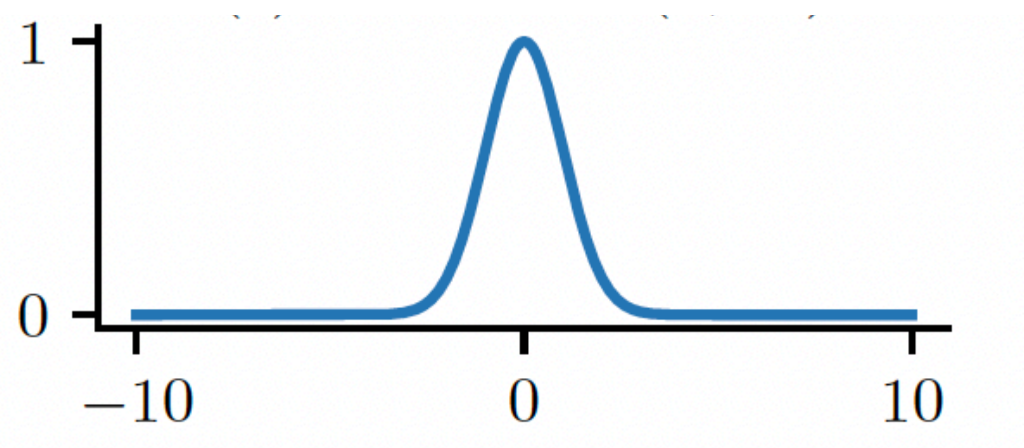


(f) RBF k(x,0.0)

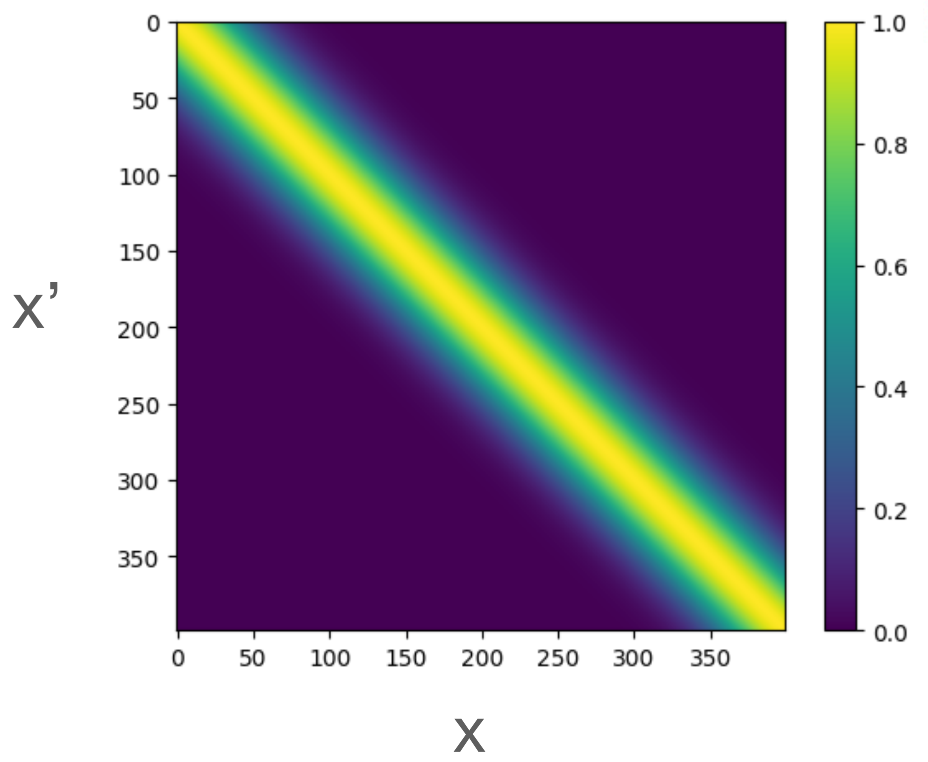
Gaussian Processes

Now consider a function $f : \mathcal{X} \to \mathbb{R}$ evaluated at a set of inputs, $\mathbf{X} = \{\boldsymbol{x}_n \in \mathcal{X}\}_{n=1}^{N}$. Let $\boldsymbol{f}_X = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]$ be the set of unknown function values at these points. If $\boldsymbol{f}_X$ is jointly Gaussian for any set of $N \geq 1$ points, then we say that $f : \mathcal{X} \to \mathbb{R}$ is a **Gaussian process**. Such a process is defined by its **mean function** $m(\boldsymbol{x}) \in \mathbb{R}$ and a **covariance function**, $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') \geq 0$, which is any positive definite **Mercer kernel**

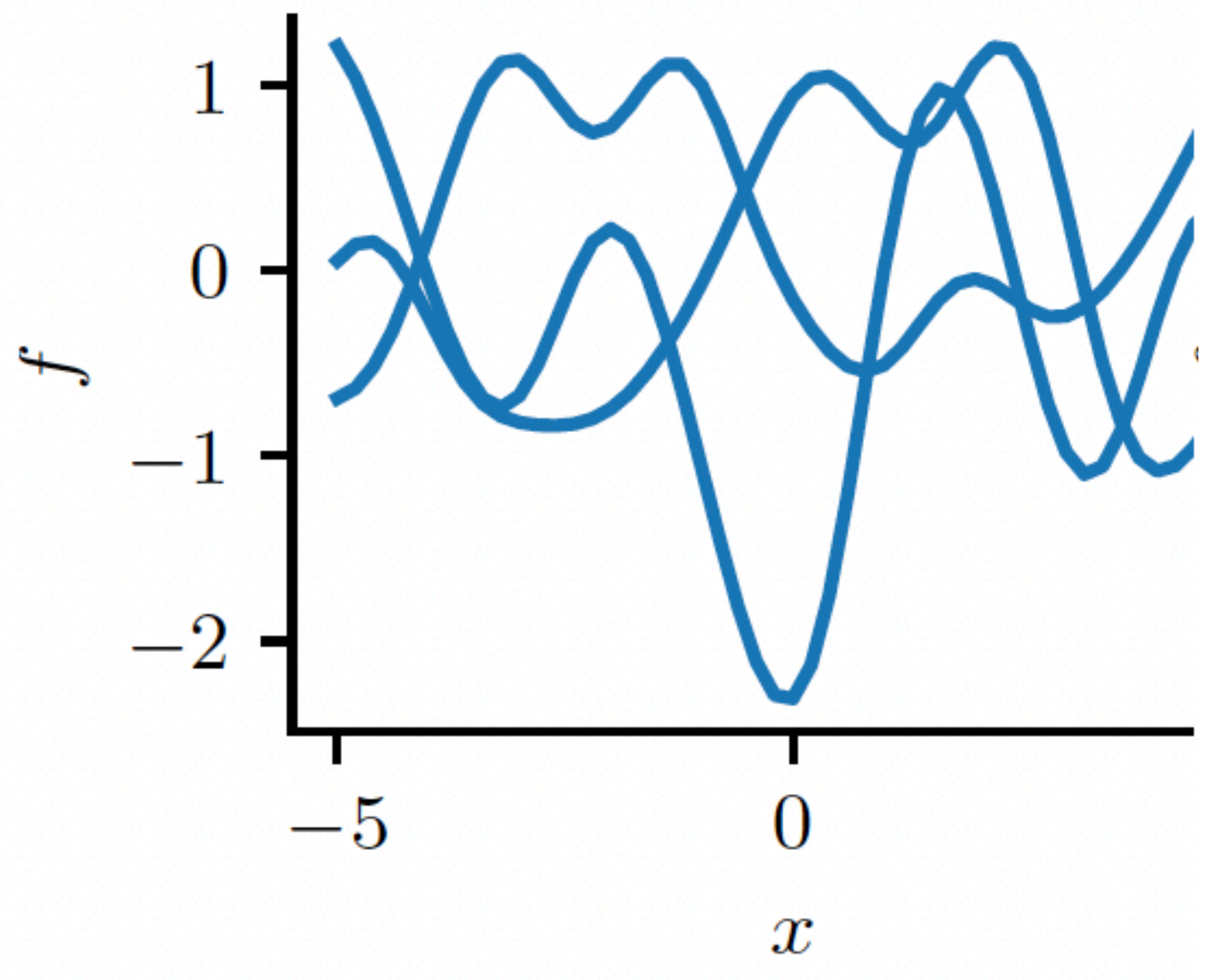
- Example Kernel ("Radial Basis Function"): $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}'; \ell) = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2\ell^2}\right)$



(f) RBF k(x,0.0)



Gaussian Processes - sampling from the prior



(a)

Figure 18.7: Left: some functions sampled from a GP prior with RBF kernel. Middle: some samples from a GP posterior, after conditioning on 5 noise-free observations. Right: some samples from a GP posterior, after conditioning on 5 noisy observations. The shaded area represents $\mathbb{E}[f(\boldsymbol{x})] \pm 2\sqrt{\mathbb{V}[f(\boldsymbol{x})]}$. *Adapted from Figure 2.2 of [RW06]. Generated by gpr_demo_noise_free.ipynb.*

Gaussian Processes - Example kernels

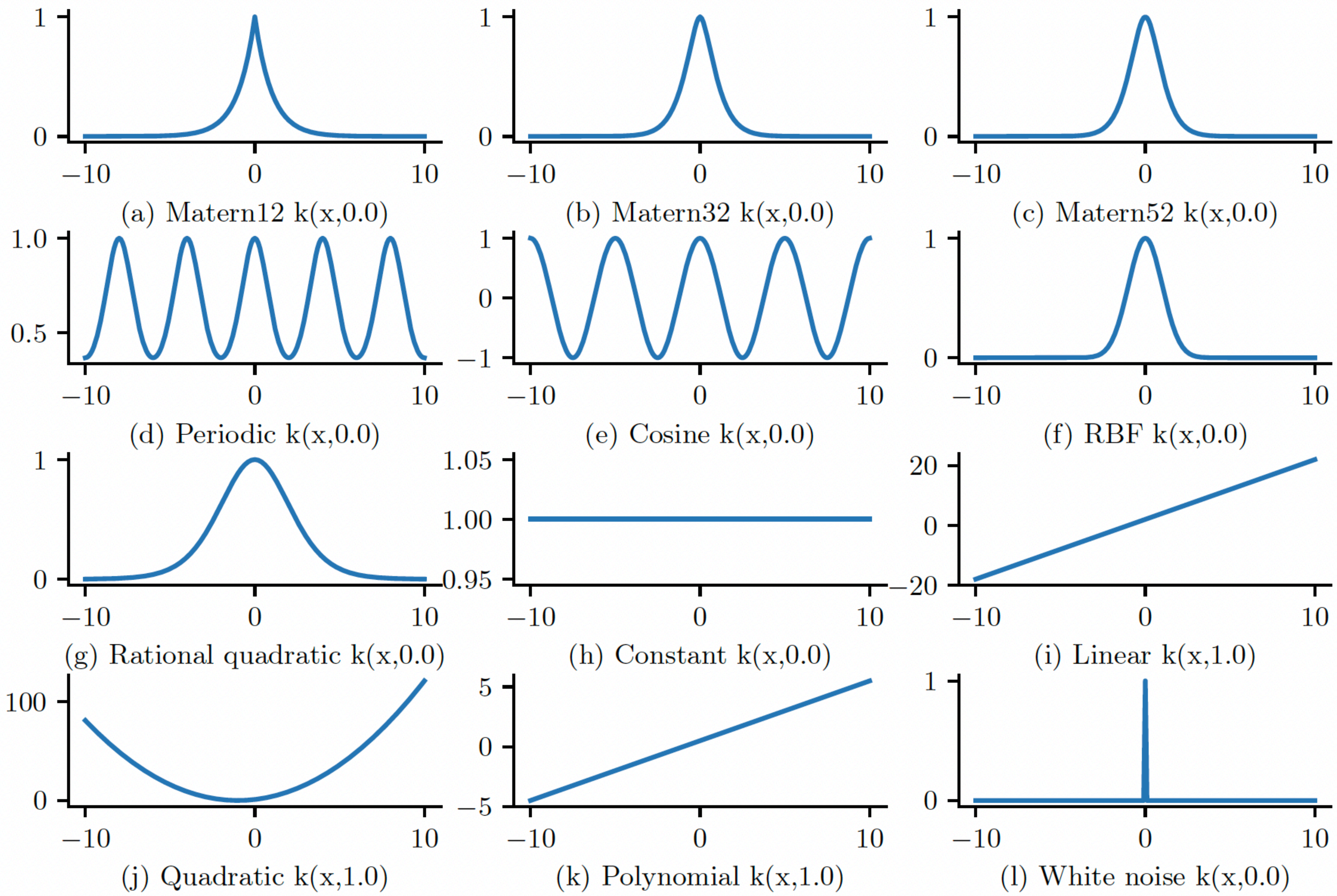


Figure 18.3: GP kernels evaluated at $k(x, 0)$ as a function of x. Generated by gpKernelPlot.ipynb.

Gaussian Processes - estimating a posterior

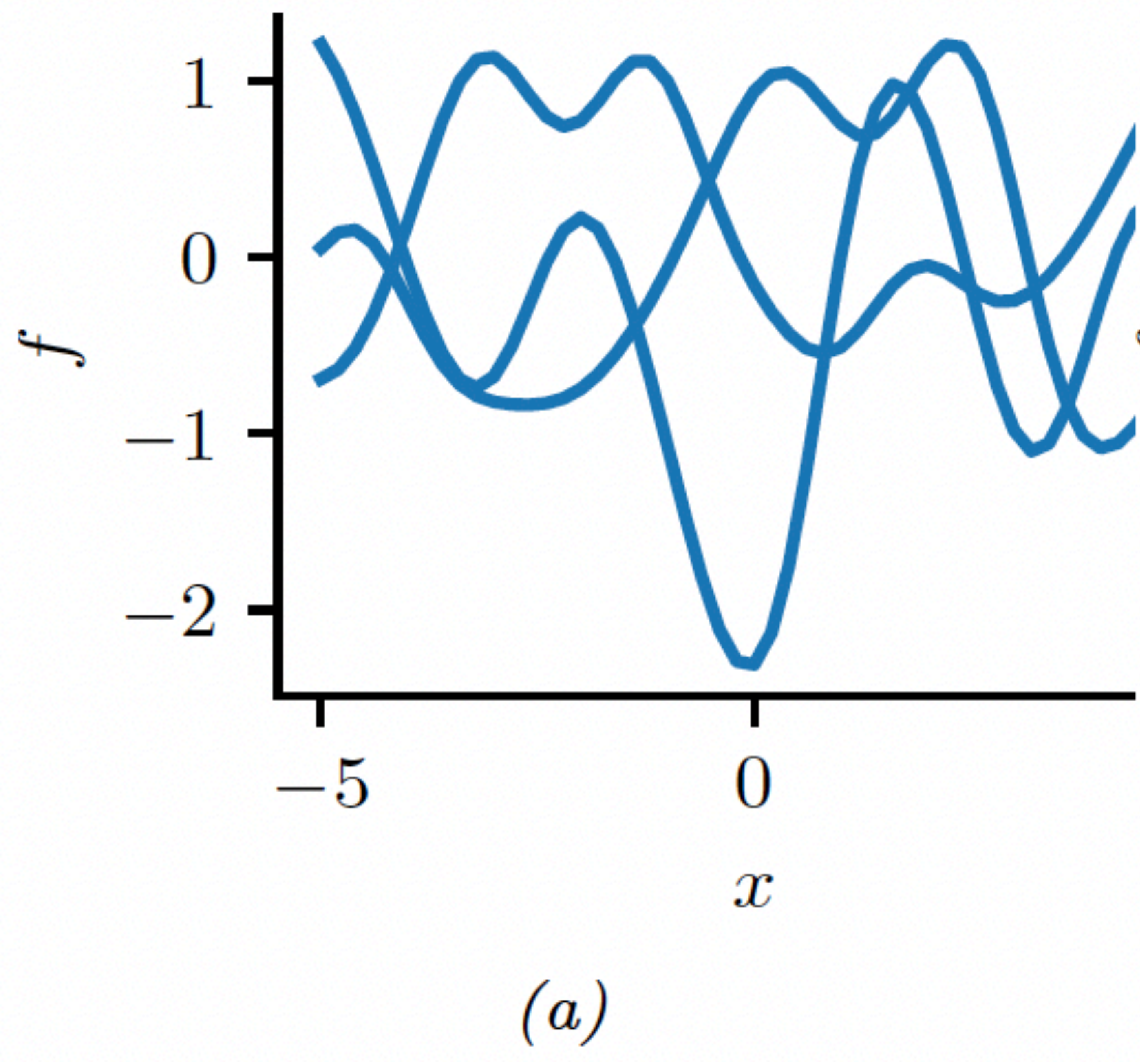


Figure 18.7: Left: some functions sampled from a GP prior with RBF kernel. Middle: some samples from a GP posterior, after conditioning on 5 noise-free observations. Right: some samples from a GP posterior, after conditioning on 5 noisy observations. The shaded area represents $\mathbb{E}[f(\boldsymbol{x})] \pm 2\sqrt{\mathbb{V}[f(\boldsymbol{x})]}$. *Adapted from Figure 2.2 of [RW06]. Generated by gpr_demo_noise_free.ipynb.*

Gaussian Processes - estimating a posterior

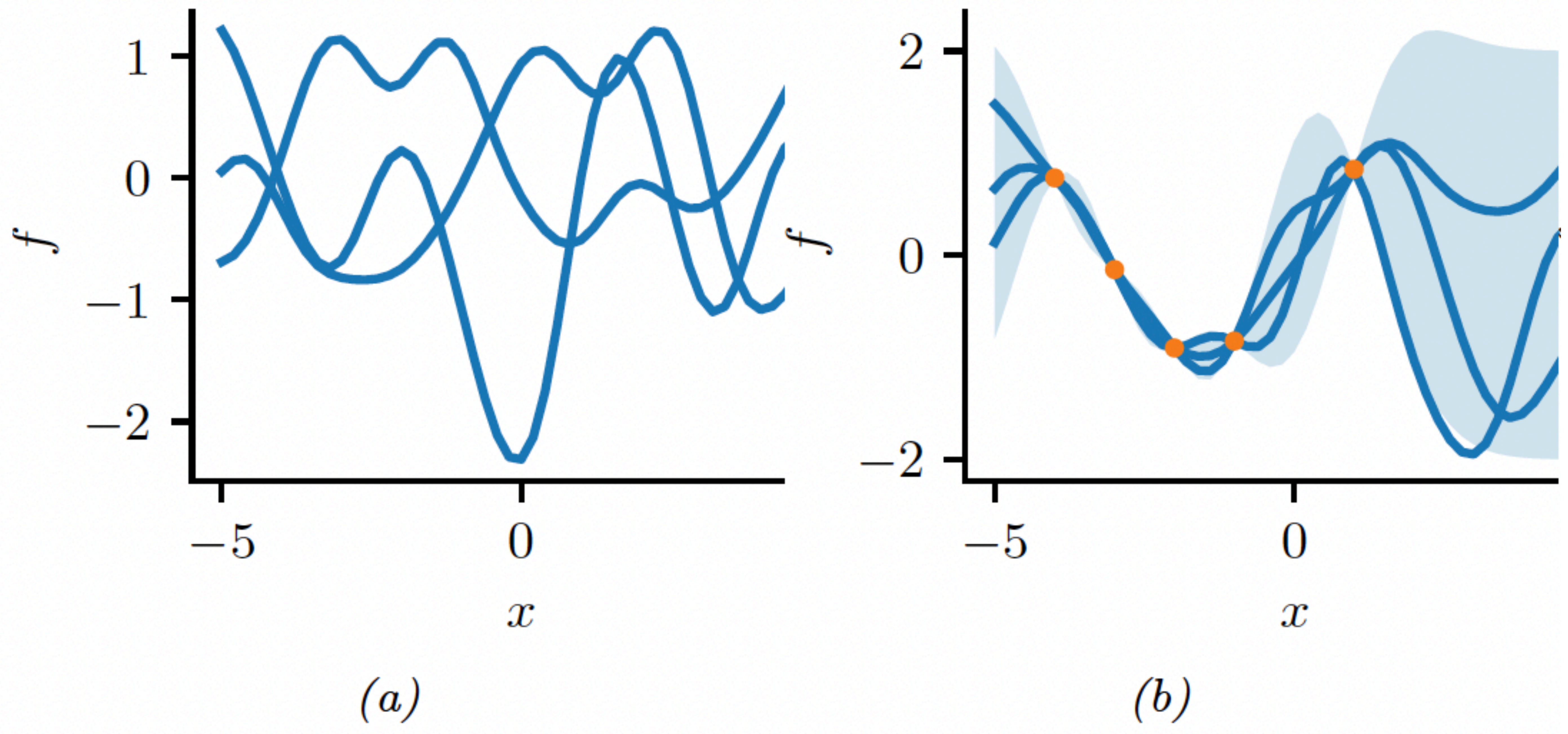


Figure 18.7: Left: some functions sampled from a GP prior with RBF kernel. Middle: some samples from a GP posterior, after conditioning on 5 noise-free observations. Right: some samples from a GP posterior, after conditioning on 5 noisy observations. The shaded area represents $\mathbb{E}[f(\boldsymbol{x})] \pm 2\sqrt{\mathbb{V}[f(\boldsymbol{x})]}$*. Adapted from Figure 2.2 of [RW06]. Generated by gpr_demo_noise_free.ipynb.*

Gaussian Processes - estimating a posterior

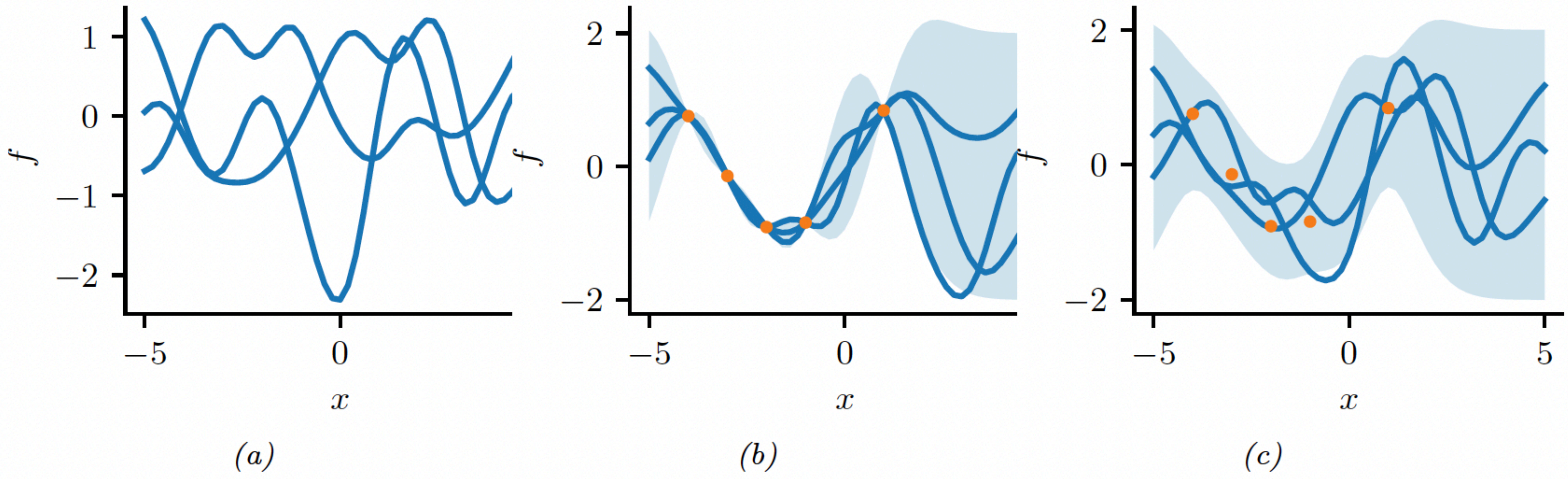


Figure 18.7: Left: some functions sampled from a GP prior with RBF kernel. Middle: some samples from a GP posterior, after conditioning on 5 noise-free observations. Right: some samples from a GP posterior, after conditioning on 5 noisy observations. The shaded area represents $\mathbb{E}[f(\boldsymbol{x})] \pm 2\sqrt{\mathbb{V}[f(\boldsymbol{x})]}$*. Adapted from Figure 2.2 of [RW06]. Generated by gpr_demo_noise_free.ipynb.*

Gaussian Process Factor Analysis (GPFA)

Yu, Cunningham, et al 2008

Gaussian Process Factor Analysis (GPFA)

- Automatically find smooth latent trajectory when inputting noisy spiking data

Yu, Cunningham, et al 2008

Gaussian Process Factor Analysis (GPFA)

$$\mathbf{y}_{:,t} \mid \mathbf{x}_{:,t} \sim \mathcal{N}\left(C\mathbf{x}_{:,t} + \mathbf{d},\ R\right),$$

Yu, Cunningham, et al 2008

Gaussian Process Factor Analysis (GPFA)

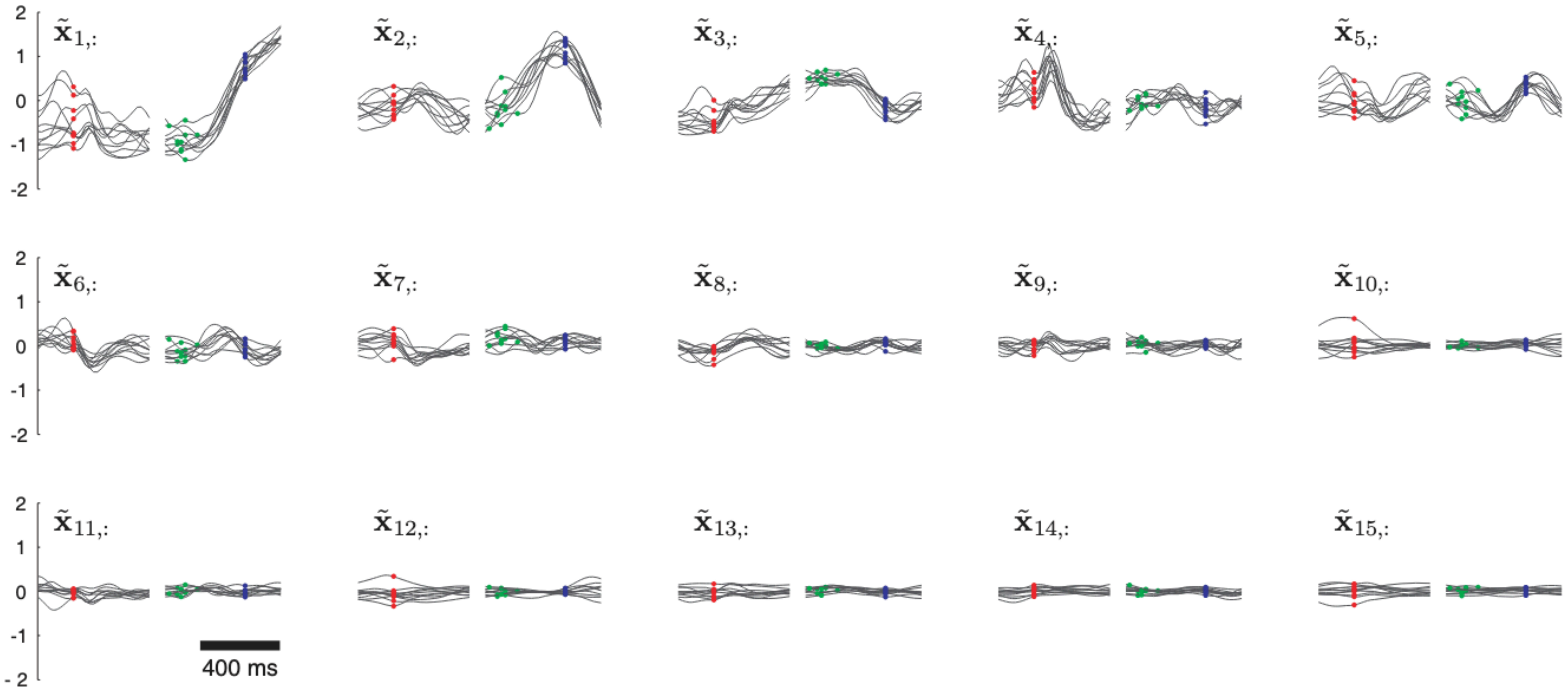
$$\mathbf{y}_{:,t} \mid \mathbf{x}_{:,t} \sim \mathcal{N}\left(C\mathbf{x}_{:,t} + \mathbf{d},\ R\right),$$

$$\mathbf{x}_{i,:} \sim \mathcal{N}\left(\mathbf{0},\ K_i\right),$$

$$K_i(t_1, t_2) = \sigma_{f,i}^2 \cdot \exp\left(-\frac{(t_1 - t_2)^2}{2 \cdot \tau_i^2}\right) + \sigma_{n,i}^2 \cdot \delta_{t_1,t_2},$$

Yu, Cunningham, et al 2008

Gaussian Process Factor Analysis (GPFA)



Yu, Cunningham, et al 2008

HMMs

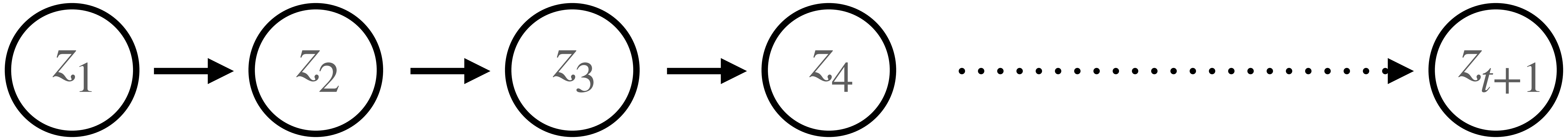
Markov Chains

$$z_1 \to z_2 \to z_3 \to z_4 \cdots\cdots \to z_{t+1}$$

- The current state only depends on the past state

$$P(z_{t+1} \mid z_1, z_2, \ldots, z_t) = P(z_{t+1} \mid z_t)$$

Markov Chains



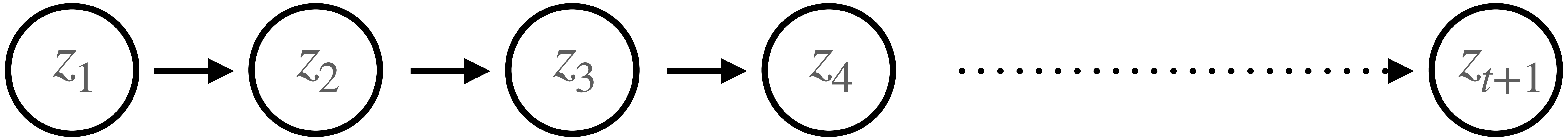
- The current state only depends on the past state

$$P(z_{t+1} \mid z_1, z_2, \ldots, z_t) = P(z_{t+1} \mid z_t)$$

- We can use the rules of independence to calculate the total probability:

$$P(z_1, z_2, \ldots, z_{t+1}) = P(z_1)P(z_2 \mid z_1) \ldots P(z_t \mid z_{t-1})P(z_{t+1} \mid z_t)$$

Markov Chains



- The current state only depends on the past state

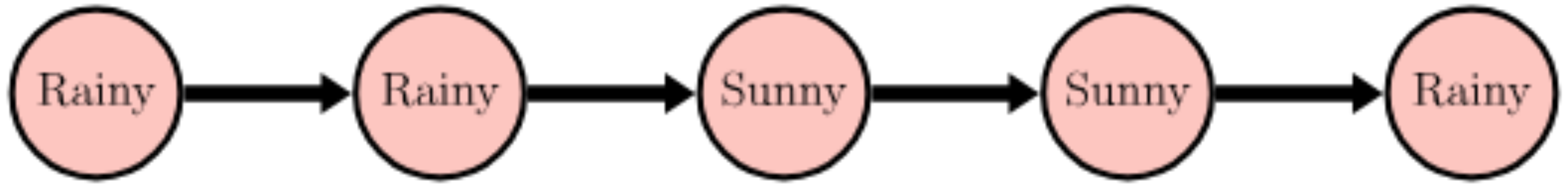
$$P(z_{t+1} \mid z_1, z_2, \ldots, z_t) = P(z_{t+1} \mid z_t)$$

- We can use the rules of independence to calculate the total probability:

$$P(z_1, z_2, \ldots, z_{t+1}) = P(z_1)P(z_2 \mid z_1) \ldots P(z_t \mid z_{t-1})P(z_{t+1} \mid z_t)$$

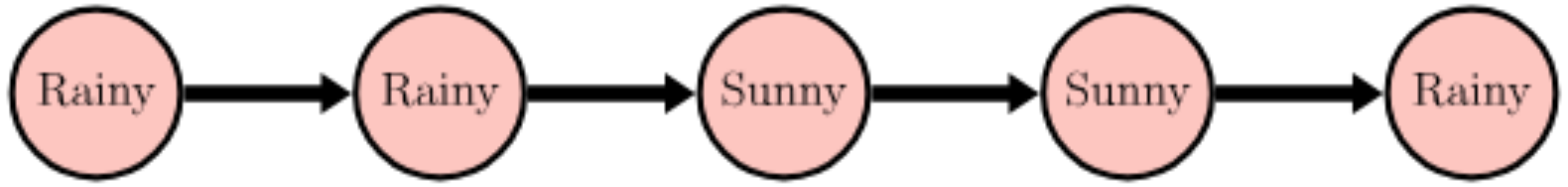
$$P(z_{1:T}) = P(z_1)\prod_{t=2}^{T} p(z_t \mid z_{t-1})$$

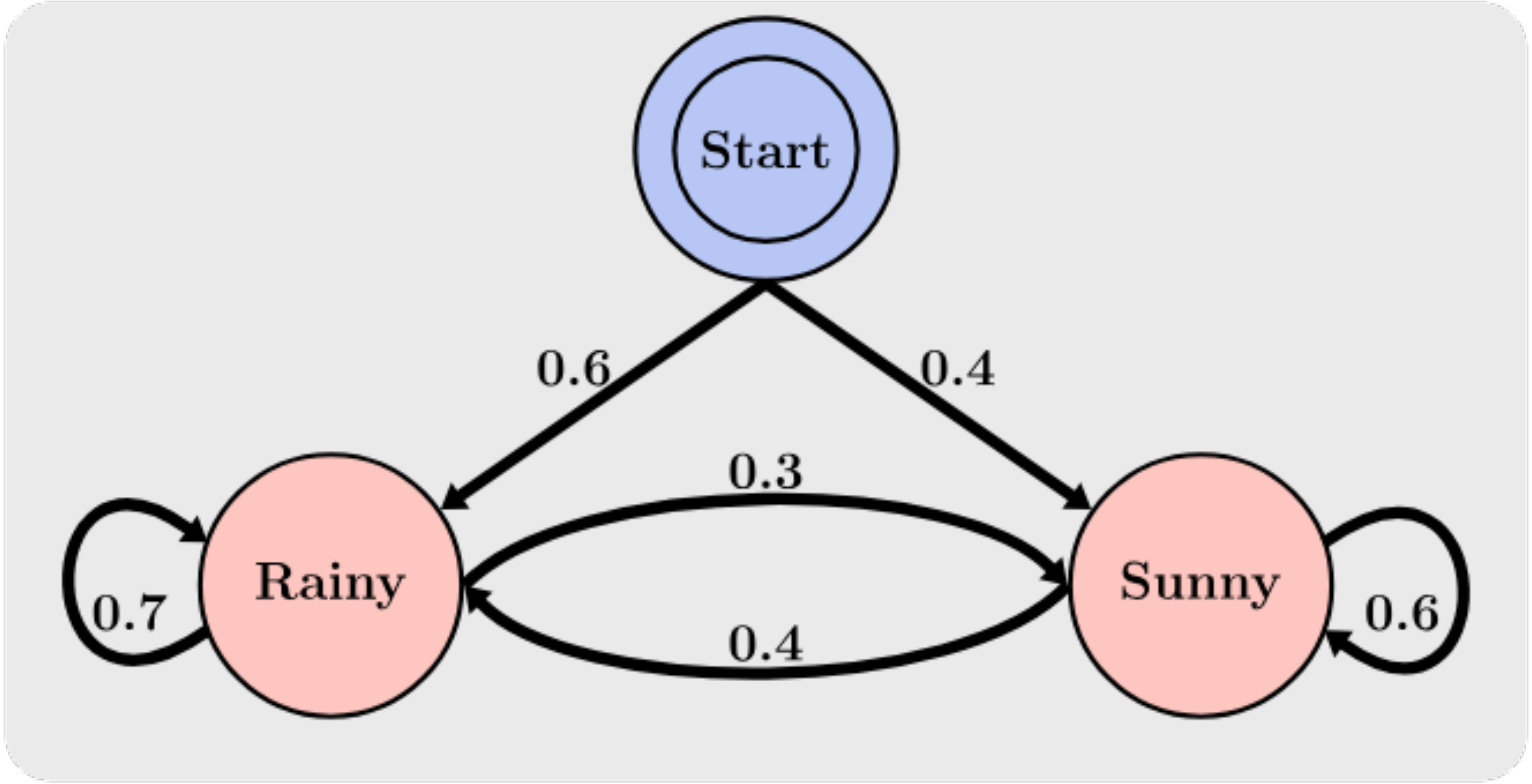
Markov Chains



https://nipunbatra.github.io/hmm/

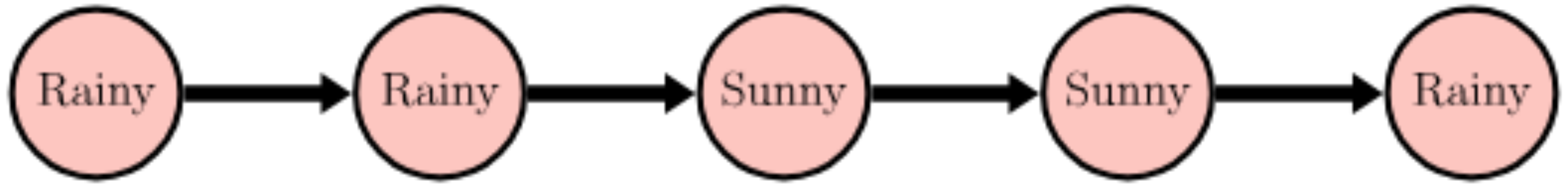
Markov Chains



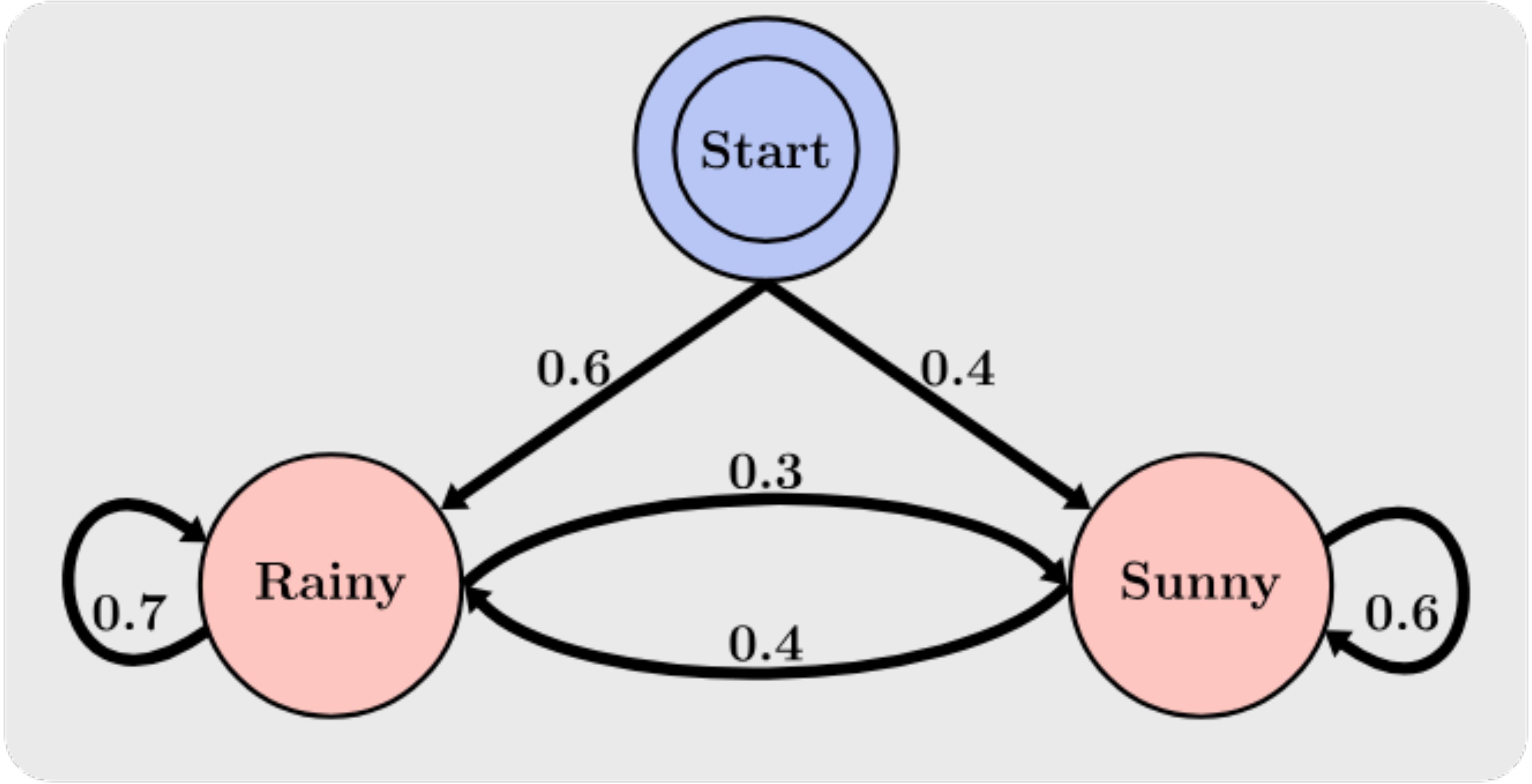


https://nipunbatra.github.io/hmm/

Markov Chains

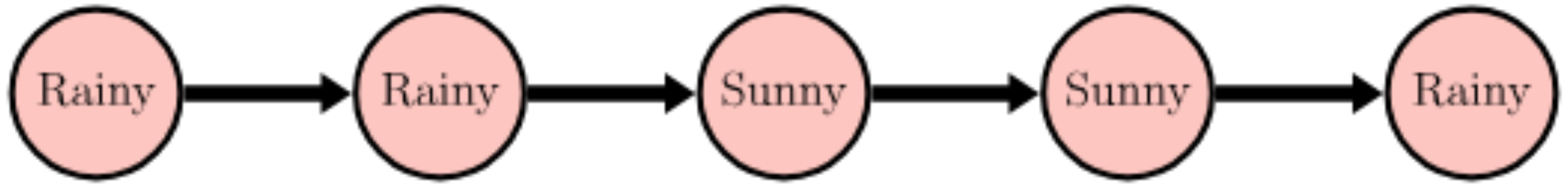


$$P(z_1, z_2, \ldots, z_{t+1}) = P(z_1)P(z_2 \mid z_1) \ldots P(z_t \mid z_{t-1})P(z_{t+1} \mid z_t)$$

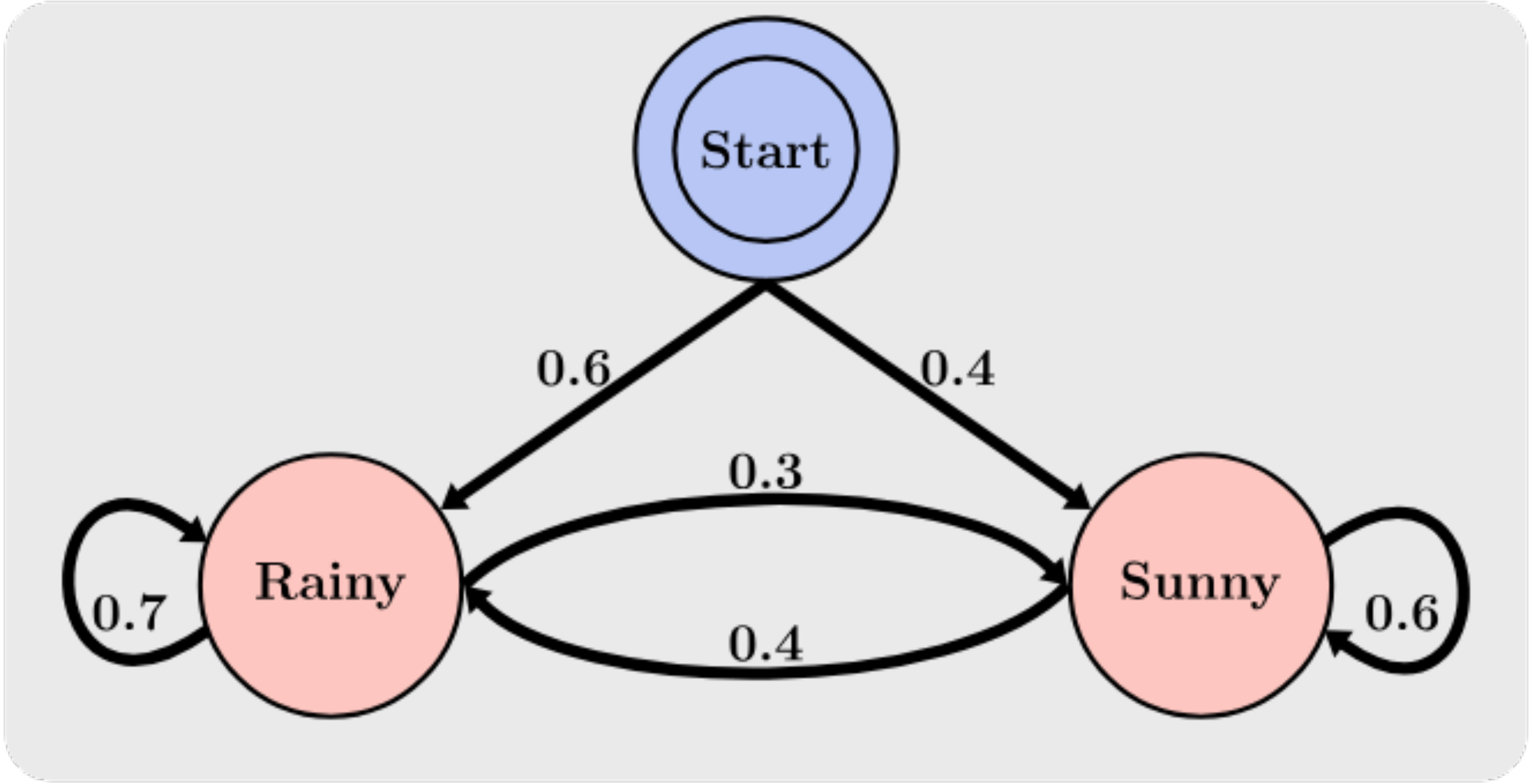


https://nipunbatra.github.io/hmm/

Markov Chains



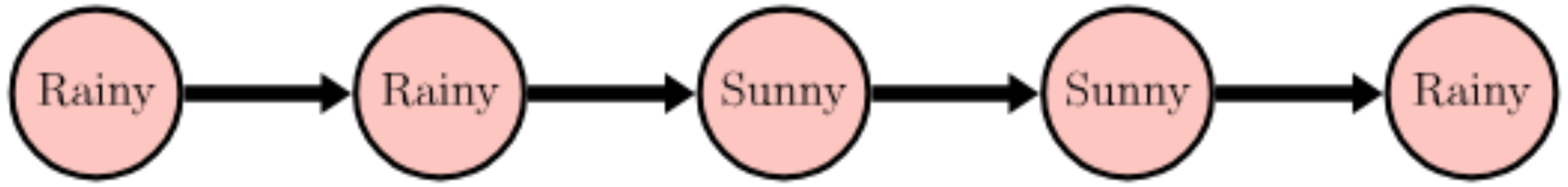
$$P(z_1, z_2, \ldots, z_{t+1}) = P(z_1)P(z_2 \mid z_1)\ldots P(z_t \mid z_{t-1})P(z_{t+1} \mid z_t)$$



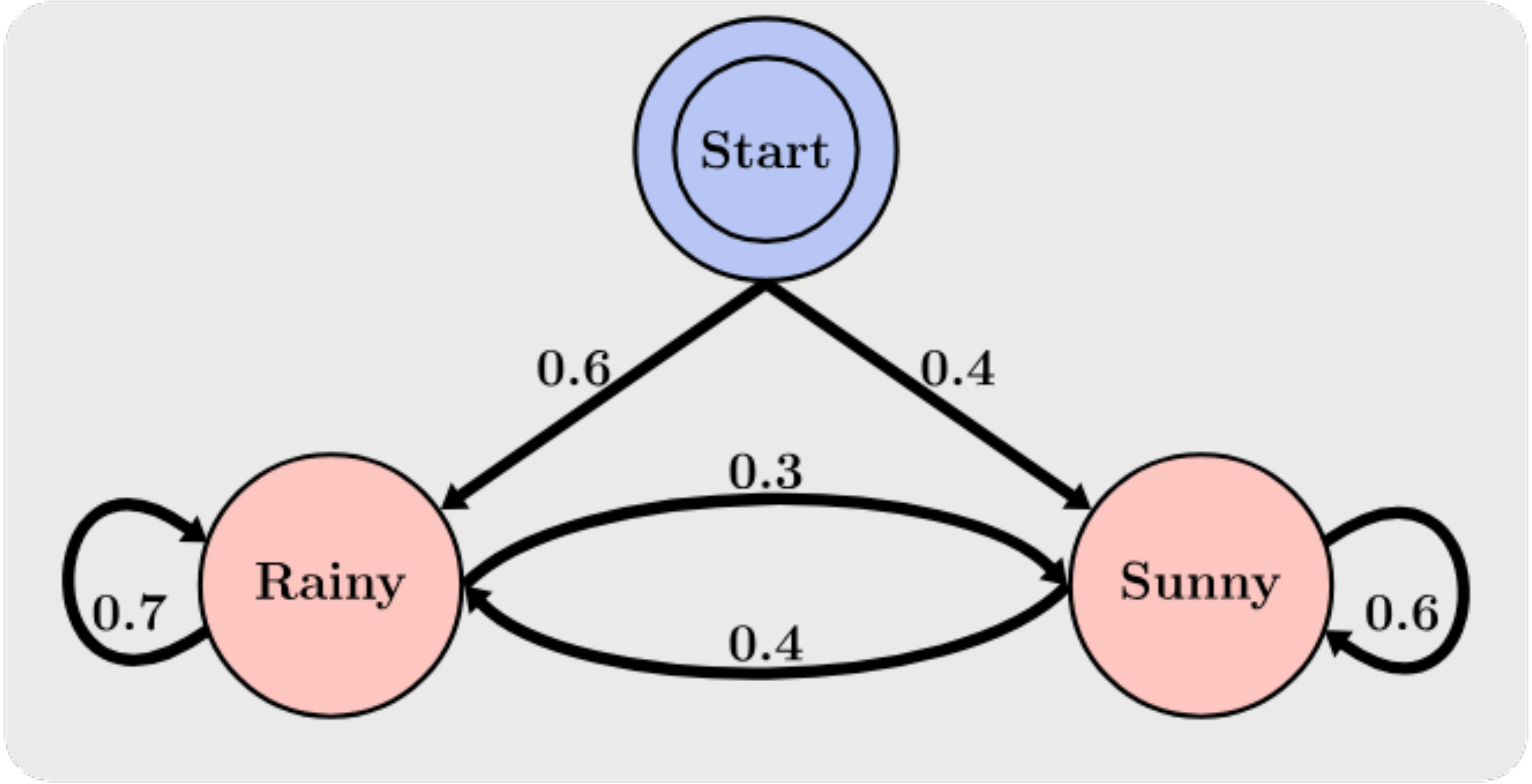
<u>Initial Conditions</u>

$$P(z_1 = R) = 0.6, \quad P(z_1 = S) = 0.4$$

Markov Chains



$$P(z_1, z_2, \ldots, z_{t+1}) = P(z_1)P(z_2 \mid z_1) \ldots P(z_t \mid z_{t-1})P(z_{t+1} \mid z_t)$$



Initial Conditions

$$P(z_1 = R) = 0.6, \quad P(z_1 = S) = 0.4$$

Transitions Matrix

$P(z_{t+1} = R \mid z_t = R) = 0.7$	$P(z_{t+1} = S \mid z_t = R) = 0.3$
$P(z_{t+1} = R \mid z_t = S) = 0.4$	$P(z_{t+1} = S \mid z_t = S) = 0.6$

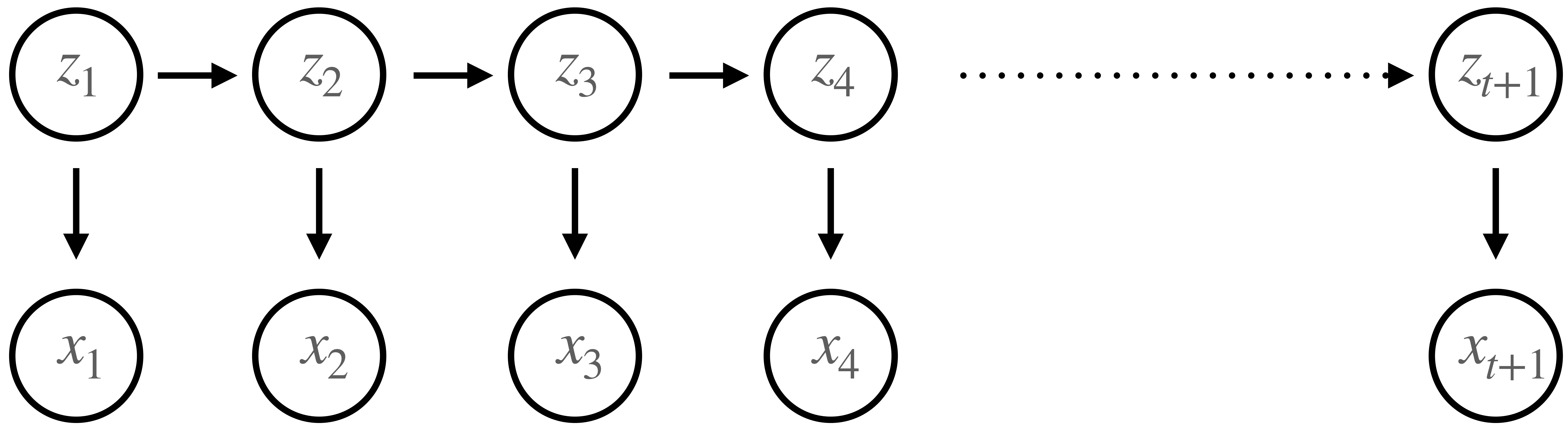
https://nipunbatra.github.io/hmm/

Hidden Markov Models

$$z_1 \rightarrow z_2 \rightarrow z_3 \rightarrow z_4 \cdots\cdots \rightarrow z_{t+1}$$

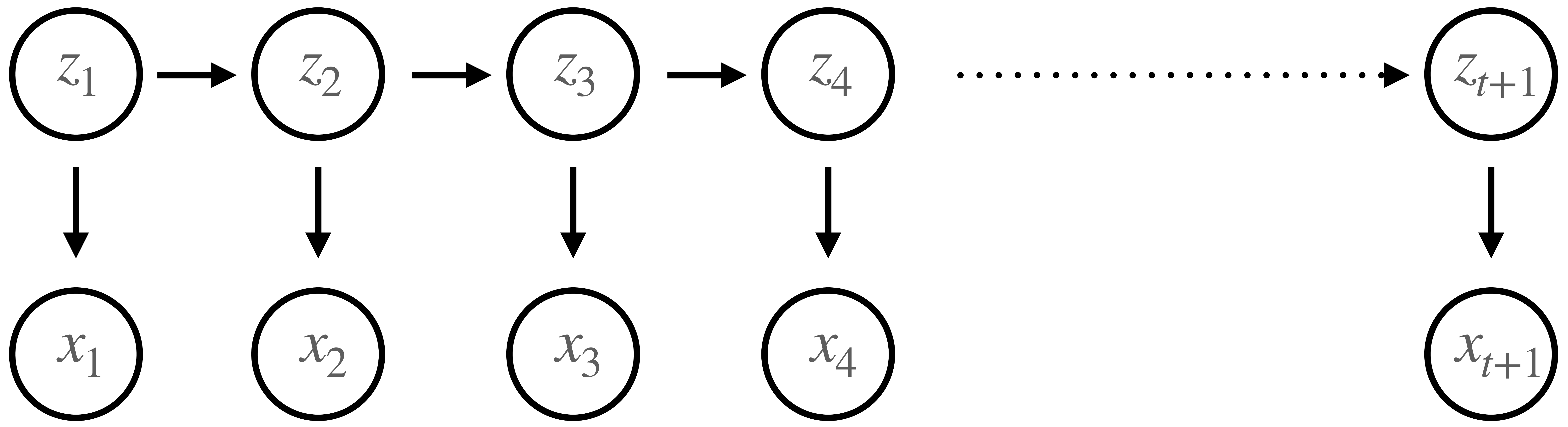
$$P(z_{1:T}) = P(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1})$$

Hidden Markov Models



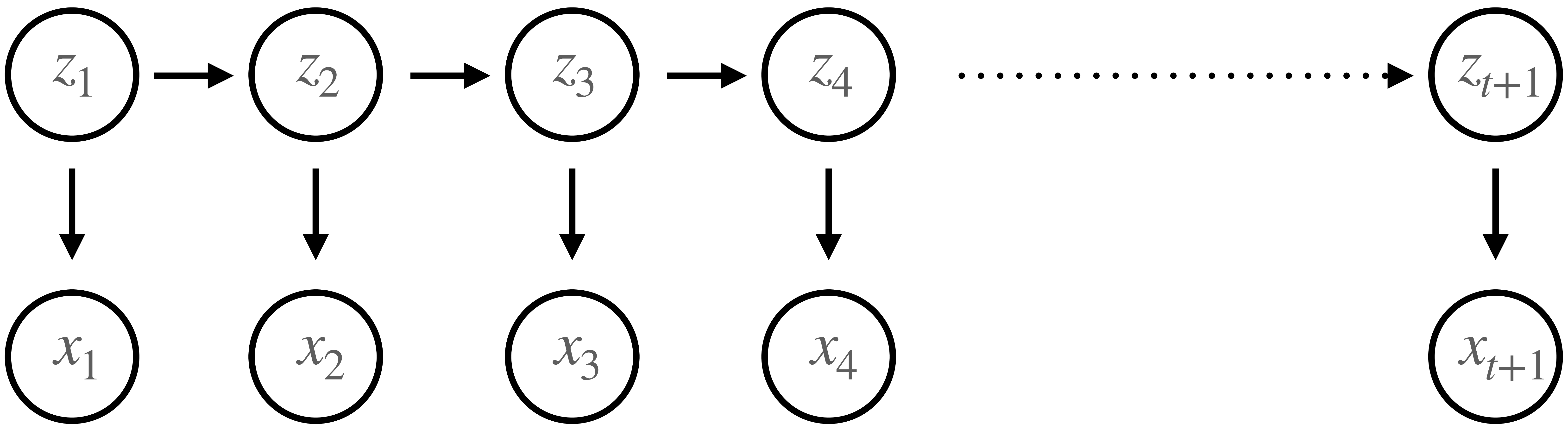
$$P(z_{1:T}) = P(z_1)\prod_{t=2}^{T} p(z_t \mid z_{t-1})$$

Hidden Markov Models



$$P(z_{1:T}, x_{1:T}) = P(z_1)\prod_{t=2}^{T} p(z_t \mid z_{t-1})\prod_{t=1}^{T} p(x_t \mid z_t)$$

Hidden Markov Models



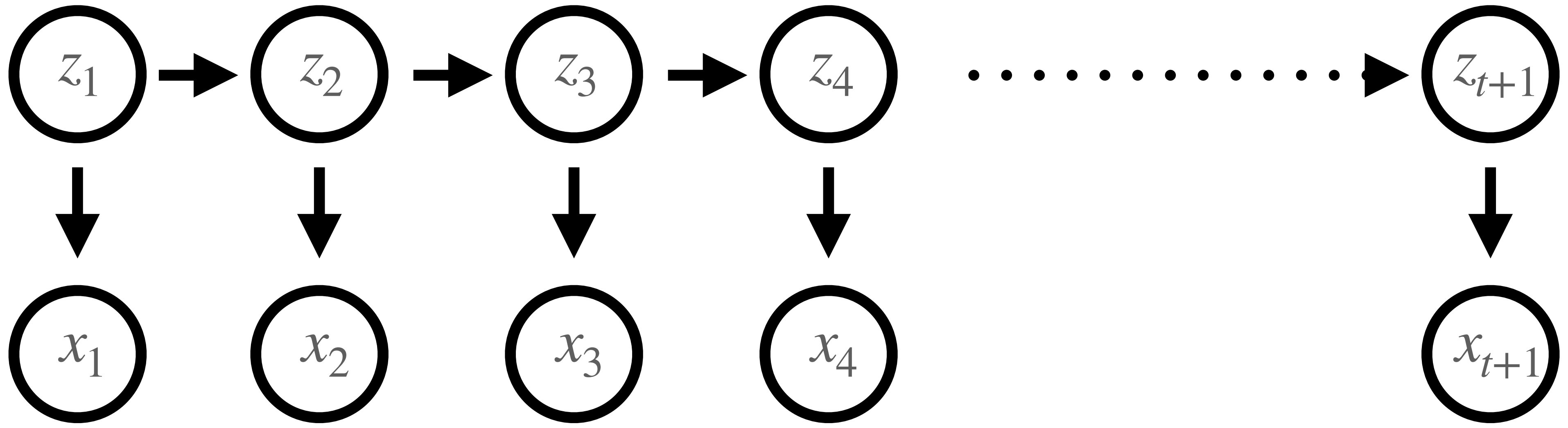
Initial Probabilities

Transition Probabilities

Emissions (Observation) Probabilities

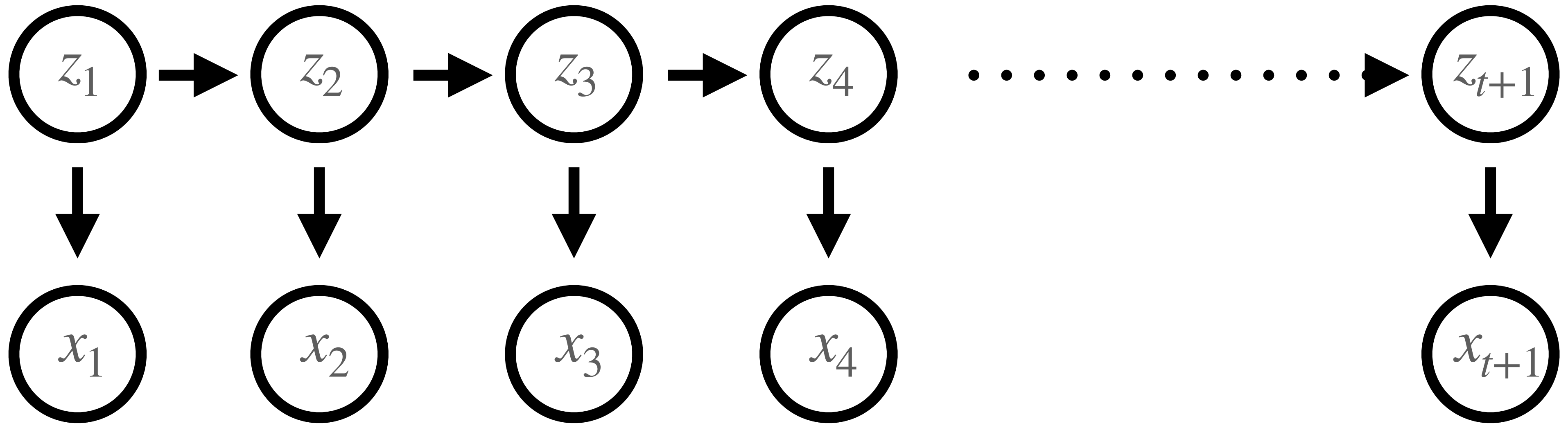
$$P(z_{1:T}, x_{1:T}) = P(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1}) \prod_{t=1}^{T} p(x_t \mid z_t)$$

Hidden Markov Models: Emissions Models



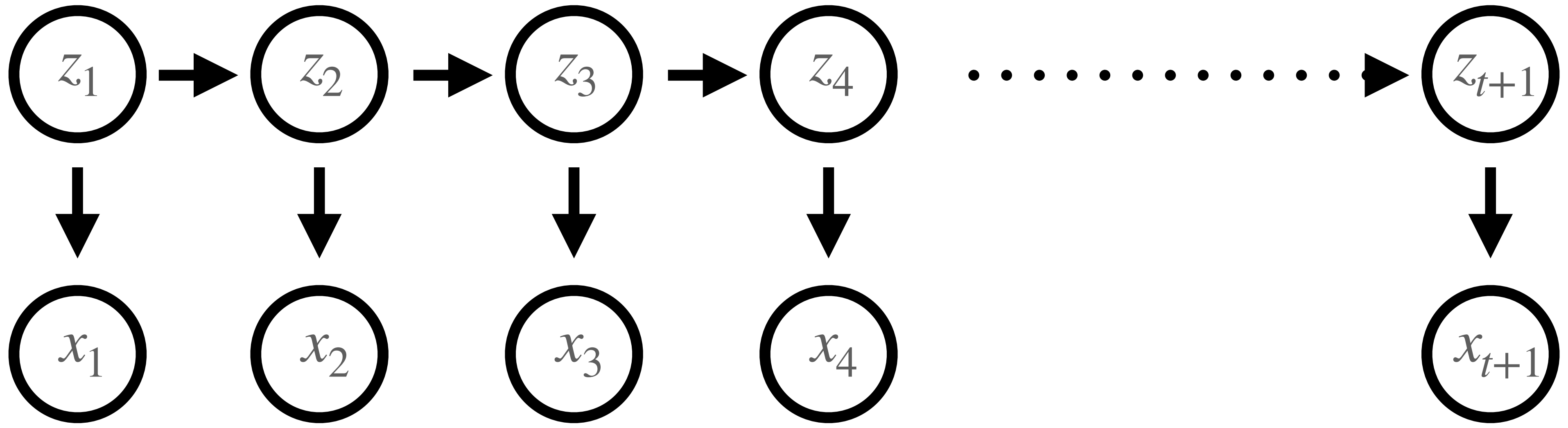
- $P(x \mid z)$ can take many different forms
  - Gaussian: $P(x_t \mid z_t) = \mathcal{N}(\mu_{z_t}, \sigma_{z_t})$

Hidden Markov Models: Emissions Models



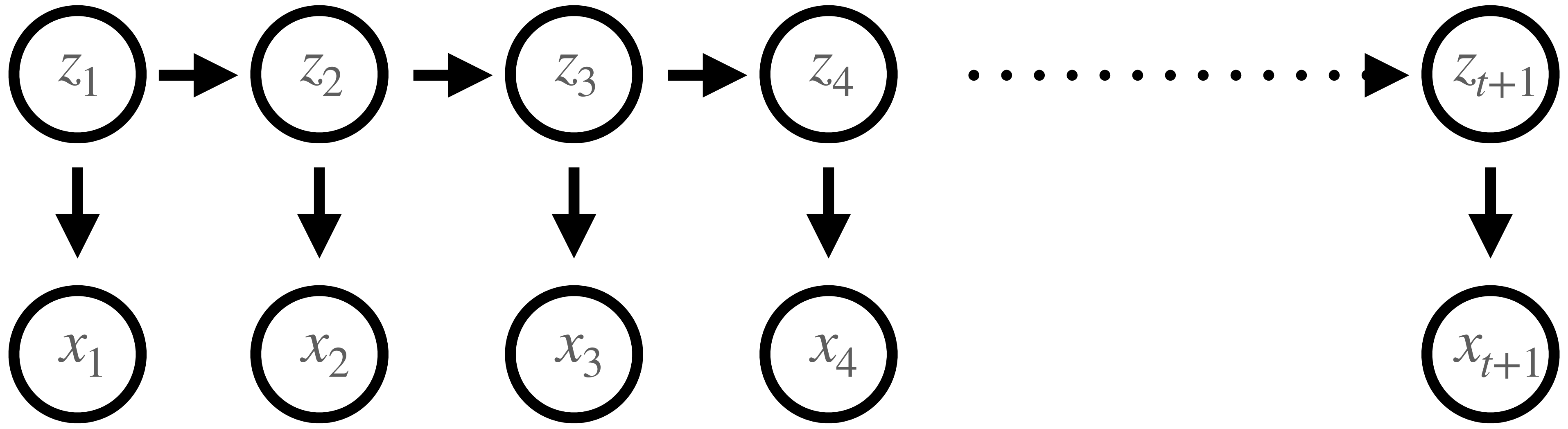
- $P(x \mid z)$ can take many different forms
  - Gaussian: $P(x_t \mid z_t) = \mathcal{N}(\mu_{z_t}, \sigma_{z_t})$
  - Bernoulli, Poisson, etc.

Hidden Markov Models: Emissions Models



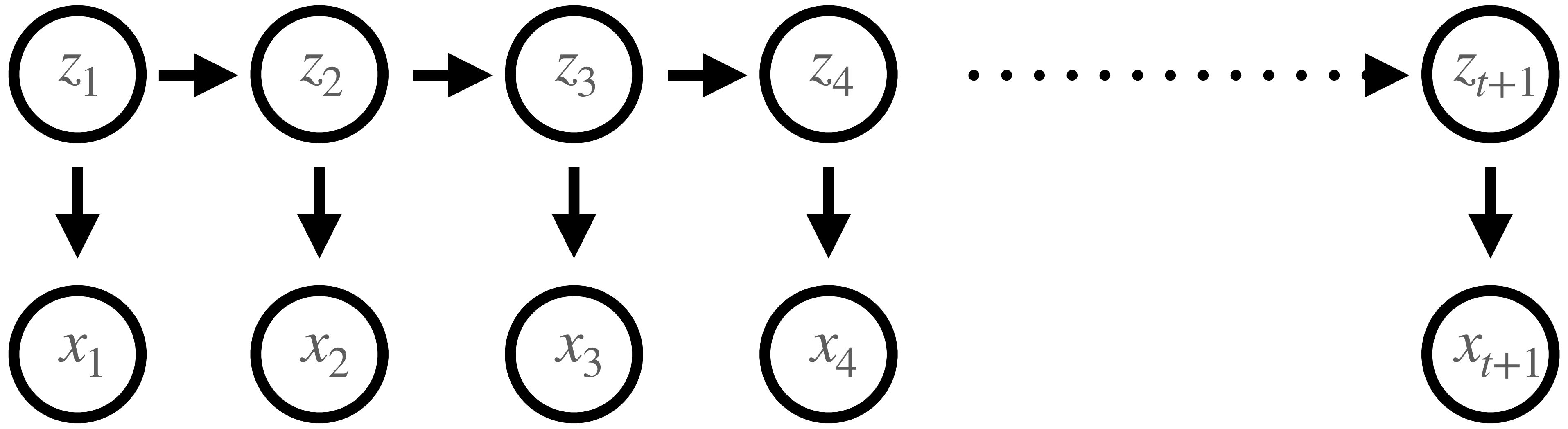
- $P(x \mid z)$ can take many different forms
  - Gaussian: $P(x_t \mid z_t) = \mathcal{N}(\mu_{z_t}, \sigma_{z_t})$
  - Bernoulli, Poisson, etc.
  - Autoregressive HMM (ARHMM):
    - Different dynamics in each discrete state: $P(x_t \mid z_t) = \mathcal{N}(y_{t-1} - A_{z_t} y_t, \sigma_{z_t})$

Hidden Markov Models: Emissions Models

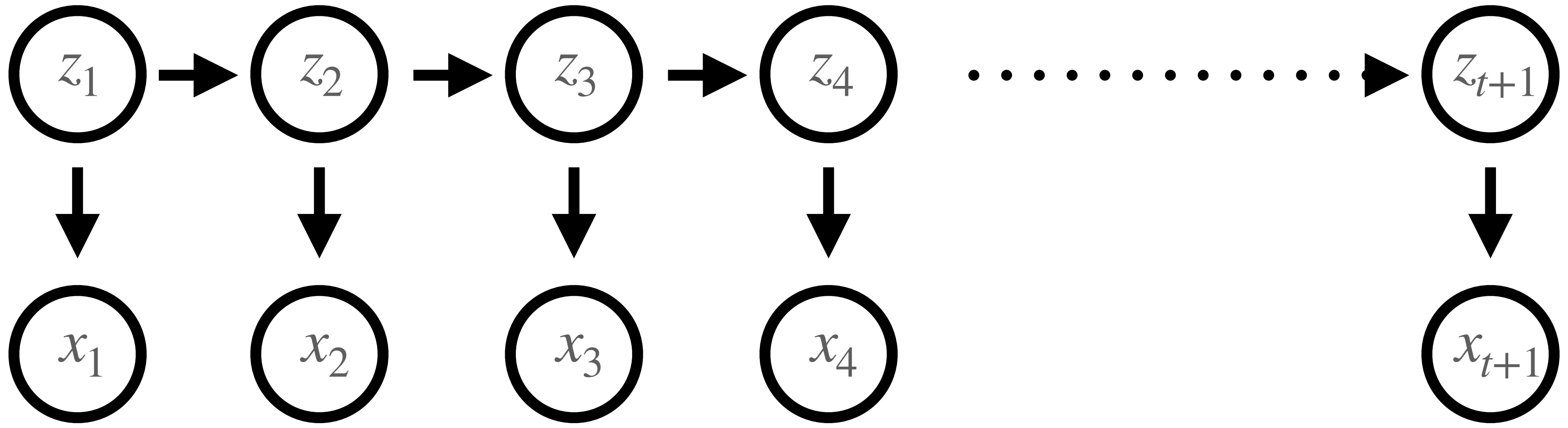


- $P(x \mid z)$ can take many different forms
    - Gaussian: $P(x_t \mid z_t) = \mathcal{N}(\mu_{z_t}, \sigma_{z_t})$
    - Bernoulli, Poisson, etc.
    - Autoregressive HMM (ARHMM):
        - Different dynamics in each discrete state: $P(x_t \mid z_t) = \mathcal{N}(y_{t-1} - A_{z_t} y_t, \sigma_{z_t})$
    - GLM-HMM:
        - Different GLM weights in each discrete state

Simulating from an HMM

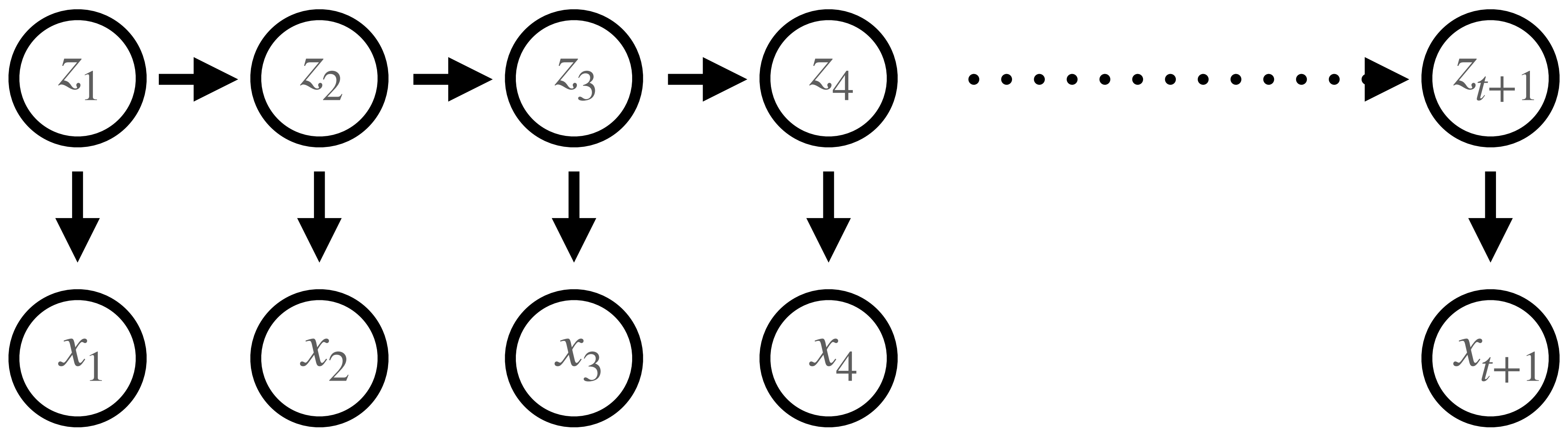


Simulating from an HMM



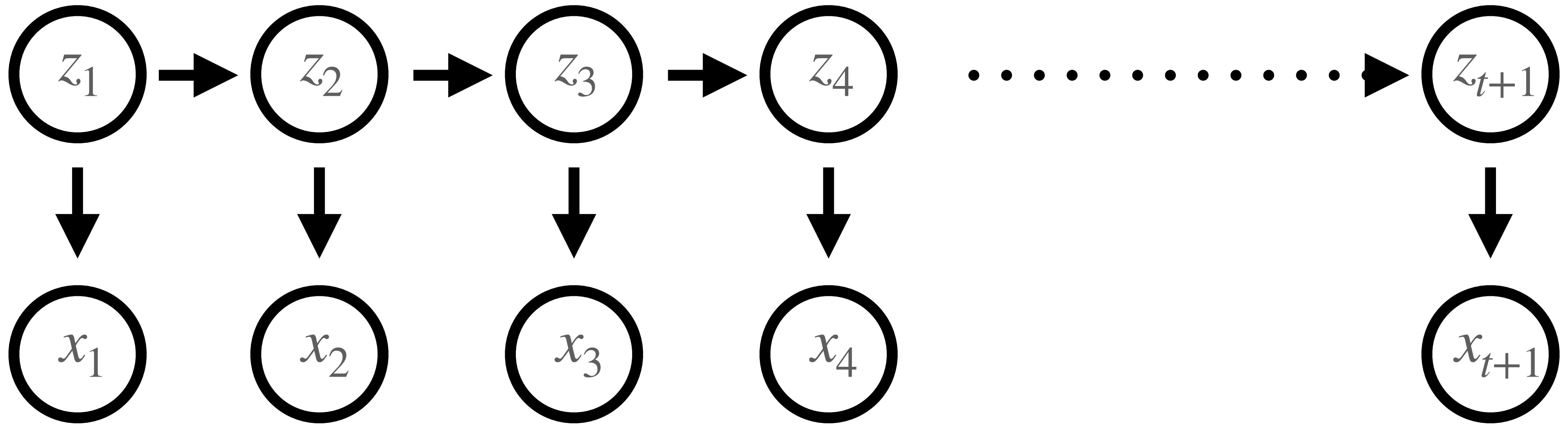
- Sample from $P(z_1)$

Simulating from an HMM



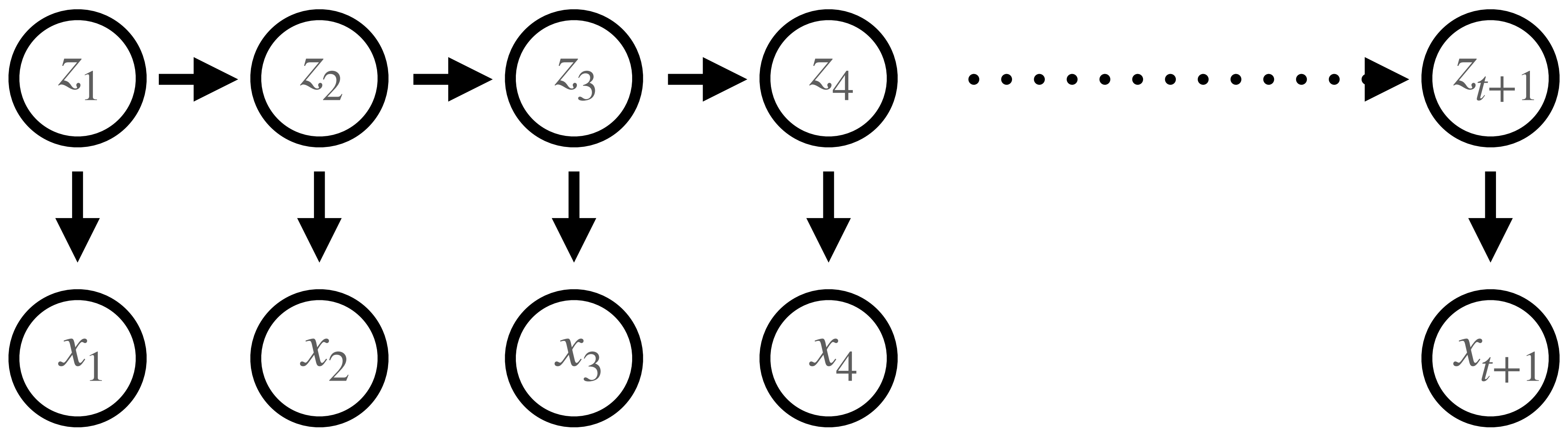
- Sample from $P(z_1)$
- Sample from $P(x_1 \mid z_1)$

Simulating from an HMM



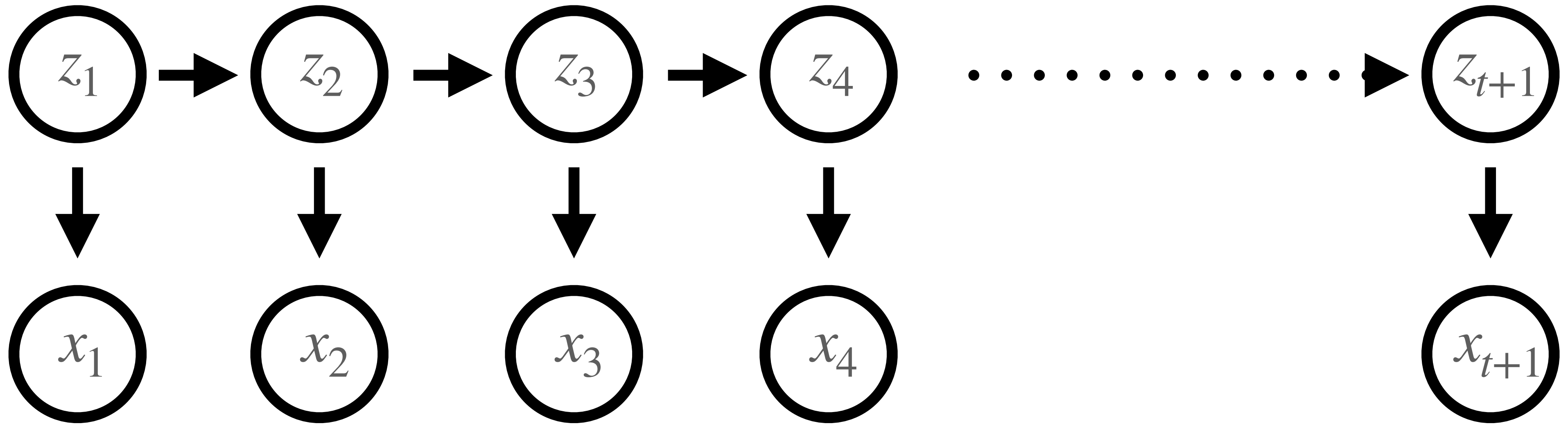
- Sample from $P(z_1)$
- Sample from $P(x_1 \mid z_1)$
- For all future time steps:
    - Sample $P(z_{t+1} \mid z_t)$
    - Sample $P(x_{t+1} \mid z_{t+1})$

HMM: Goals



- Given some data:
  - Fit the model!
    - Infer discrete latent states with Forward/Backward Algorithm
    - Infer model parameters (transition probabilities, emissions model)
  - Determine the likelihood of the data given the model parameters

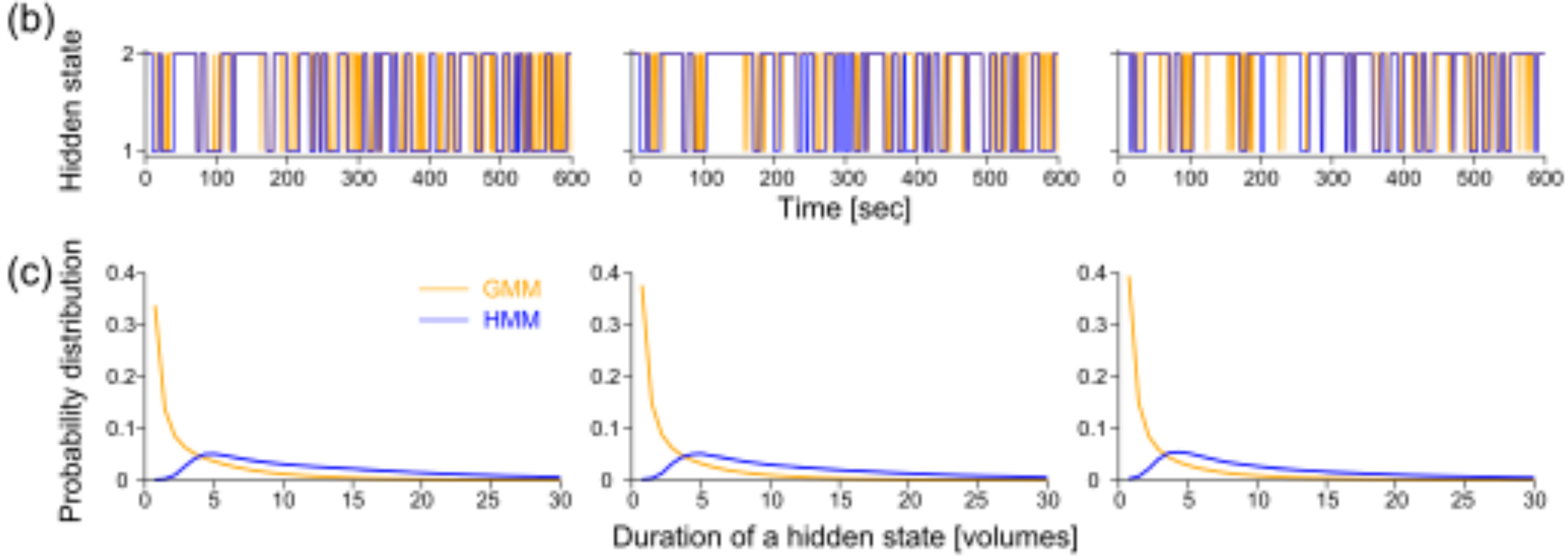
Hidden Markov Models: Emissions Models



- $P(x \mid z)$ can take many different forms
  - Gaussian: $P(x_t \mid z_t) = \mathcal{N}(\mu_{z_t}, \sigma_{z_t})$
  - Bernoulli, Poisson, etc.
  - Autoregressive HMM (ARHMM):
    - Different dynamics in each discrete state: $P(x_t \mid z_t) = \mathcal{N}(y_{t-1} - A_{z_t} y_t, \sigma_{z_t})$
  - GLM-HMM:
    - different GLM weights in each discrete state

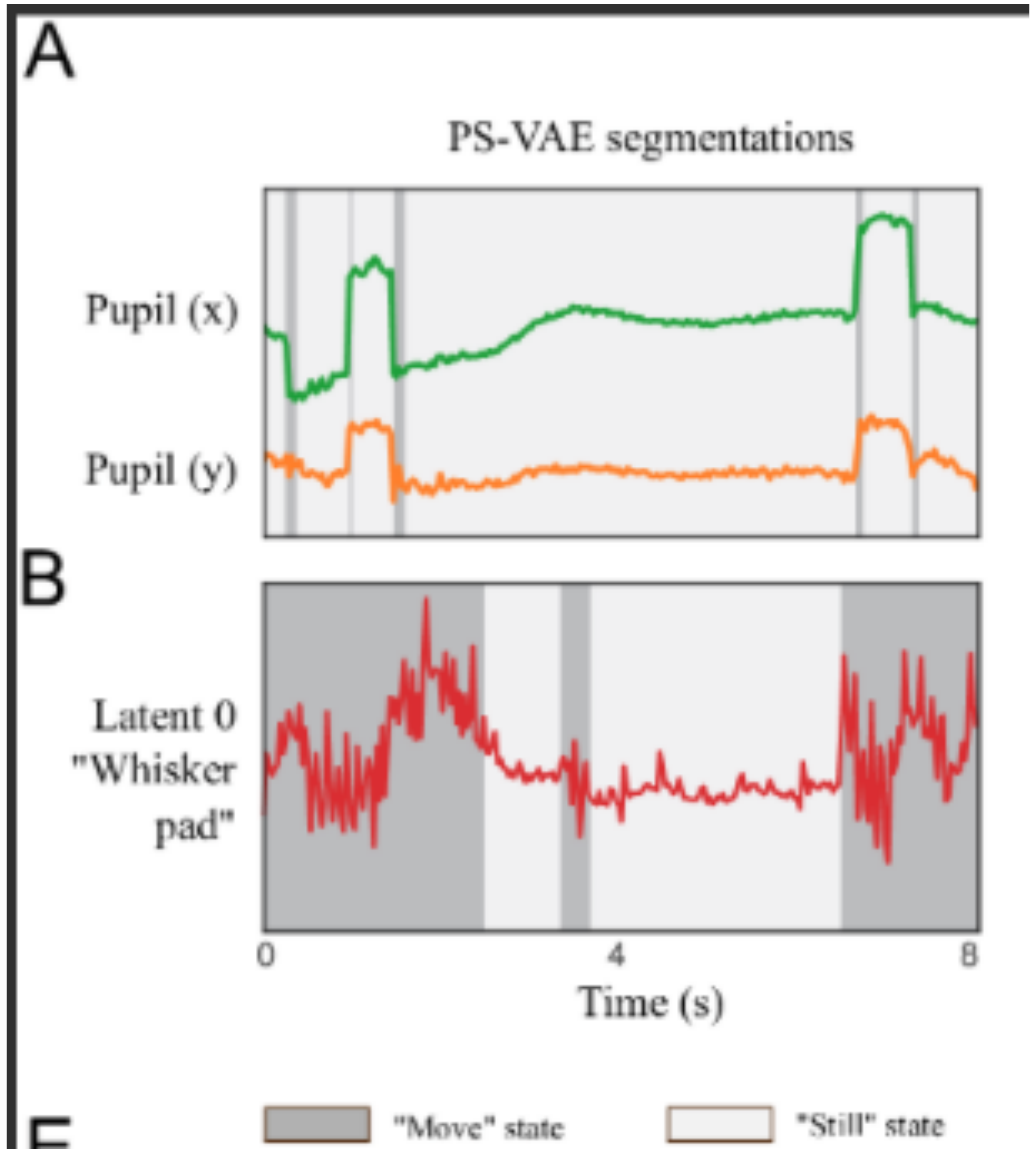
HMM - Gaussian Emissions

Modelling state-transition dynamics in resting-state brain signals by the hidden Markov and Gaussian mixture models



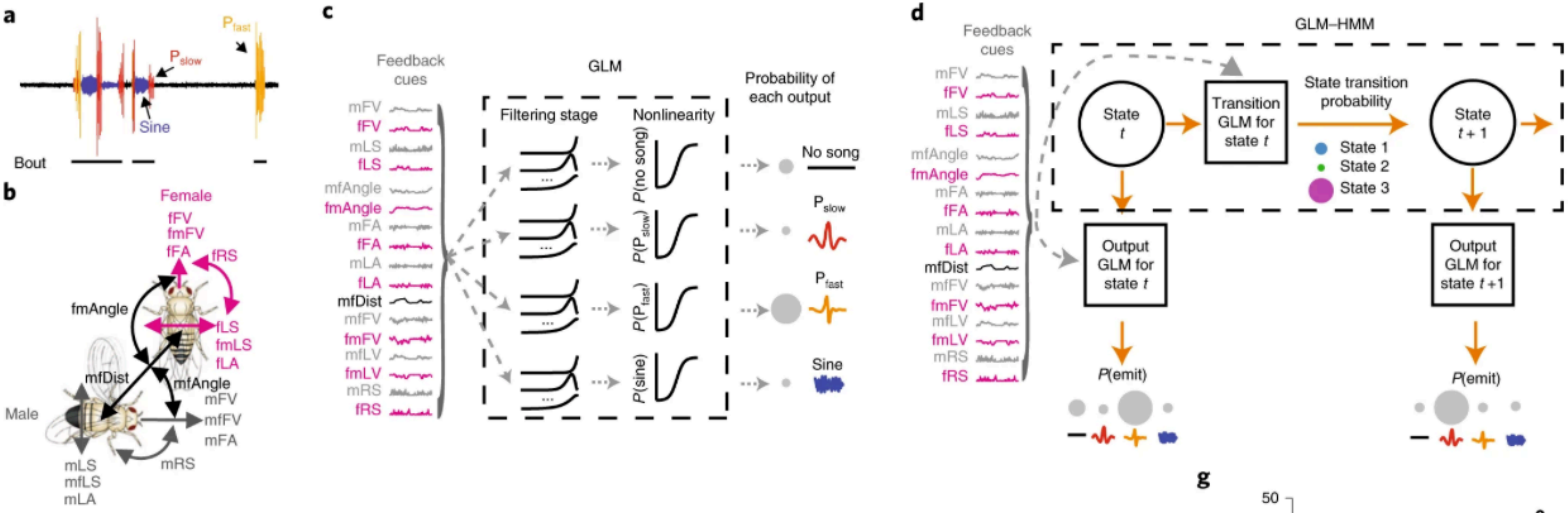
HMM - Autoregressive Emissions

Partitioning variability in animal behavioral videos using semi-supervised variational autoencoders



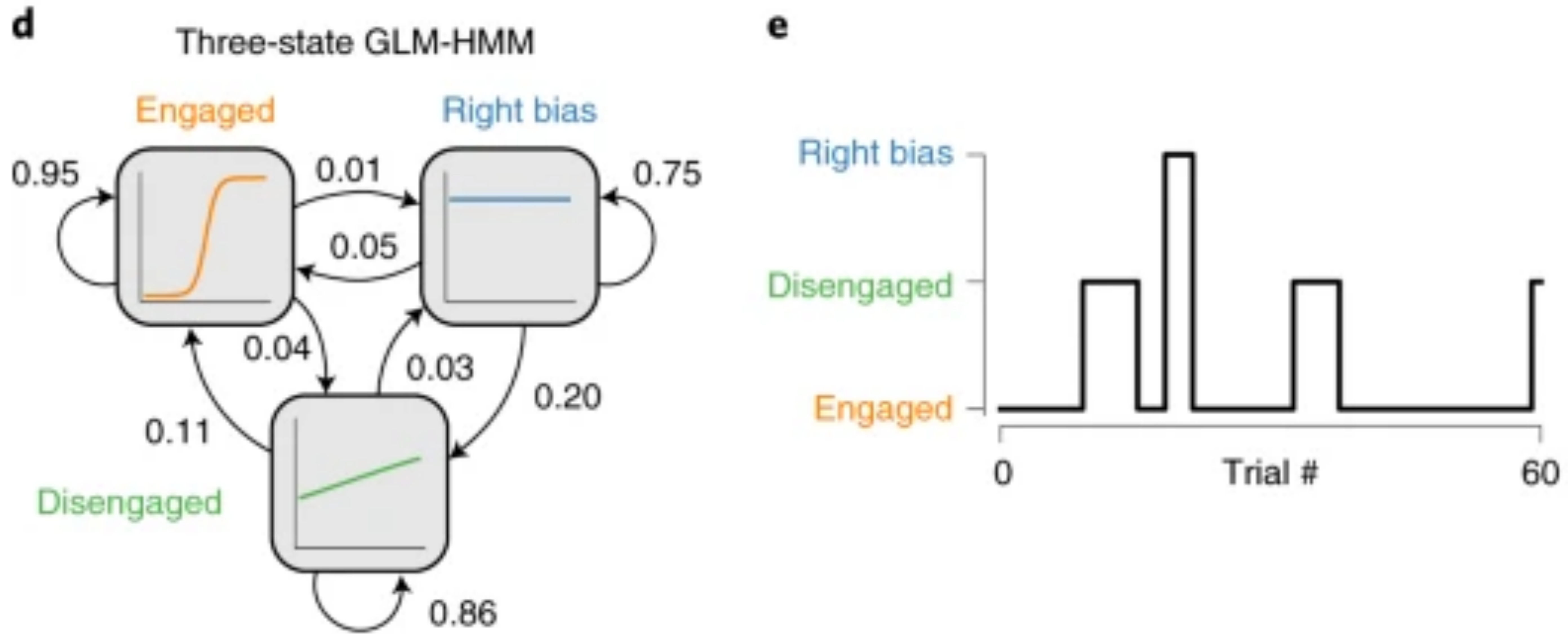
HMM - GLM Emissions

Unsupervised identification of the internal states that shape natural behavior



HMM - GLM Emissions

Mice alternate between discrete strategies during perceptual decision-making



Resources

- Probabilistic Machine Learning Book 1
  - https://probml.github.io/pml-book/book1.html
- Intro to LVM Notes from Princeton Course
  - https://pillowlab.princeton.edu/teaching/statneuro2020/notes/notes18_LatentVariableModels.pdf
- Interactive HMM Website
  - https://nipunbatra.github.io/hmm/