Latent Variable Models for Sequential/Time-Series Data

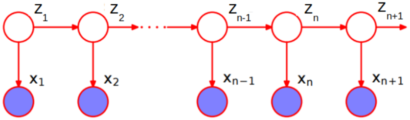
Piyush Rai

Probabilistic Machine Learning (CS772A)

Nov 7, 2017

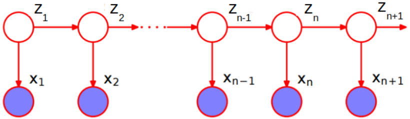
Latent Variable Models for Sequential Data

- Task: Given a sequence of observations, infer the latent state of each observation

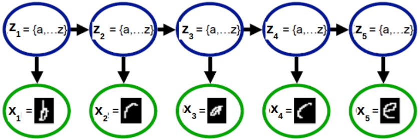


Latent Variable Models for Sequential Data

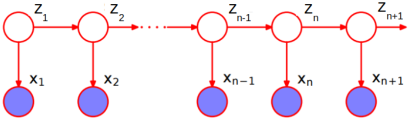
- Task: Given a sequence of observations, infer the latent state of each observation



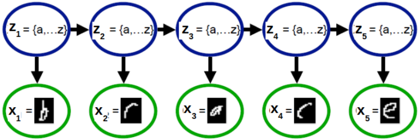
- An example: Recognizing a sequence of handwritten characters



- Task: Given a sequence of observations, infer the latent state of each observation



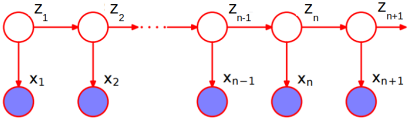
- An example: Recognizing a sequence of handwritten characters



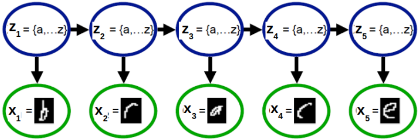
  - In this example, the latent state z_n at step n is a discrete value

Latent Variable Models for Sequential Data

- Task: Given a sequence of observations, infer the latent state of each observation



- An example: Recognizing a sequence of handwritten characters

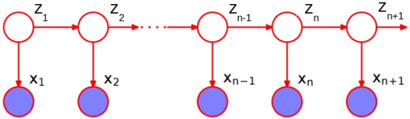


  - In this example, the latent state $\boldsymbol{z}_n$ at step n is a discrete value

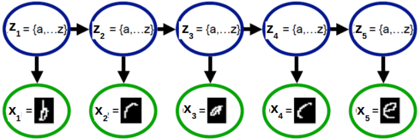
- Another example: Given a sequence of observed noisy 2D coordinates $\boldsymbol{x}_n$ of an object, infer its latent state $\boldsymbol{z}_n$, e.g., actual coordinates, velocity, acceleration, etc. at each step $n = 1, 2, \ldots$

Latent Variable Models for Sequential Data

- Task: Given a sequence of observations, infer the latent state of each observation



- An example: Recognizing a sequence of handwritten characters



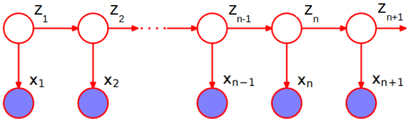
  - In this example, the latent state $\boldsymbol{z}_n$ at step n is a discrete value

- Another example: Given a sequence of observed noisy 2D coordinates $\boldsymbol{x}_n$ of an object, infer its latent state $\boldsymbol{z}_n$, e.g., actual coordinates, velocity, acceleration, etc. at each step $n = 1, 2, \ldots$
  - In this example, the latent state $\boldsymbol{z}_n$ at step n is a continuous vector

Latent Variable Models for Sequential Data

- For both cases (discrete/continuous $\boldsymbol{z}_n$), the generic model can be written as follows

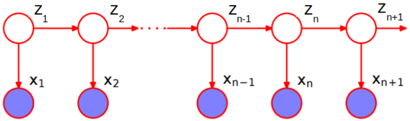
$$\begin{aligned} \boldsymbol{z}_n|\boldsymbol{z}_{n-1} &\sim p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) && \text{(first-order dependence b/w } \boldsymbol{z}_n\text{'s)} \\ \boldsymbol{x}_n|\boldsymbol{z}_n &\sim p(\boldsymbol{x}_n|\boldsymbol{z}_n) && \text{(i.i.d. draws of } \boldsymbol{x}_n \text{ given } \boldsymbol{z}_n\text{)} \end{aligned}$$



Latent Variable Models for Sequential Data

- For both cases (discrete/continuous $\boldsymbol{z}_n$), the generic model can be written as follows

$$\begin{aligned} \boldsymbol{z}_n|\boldsymbol{z}_{n-1} &\sim p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) && \text{(first-order dependence b/w } \boldsymbol{z}_n\text{'s)} \\ \boldsymbol{x}_n|\boldsymbol{z}_n &\sim p(\boldsymbol{x}_n|\boldsymbol{z}_n) && \text{(i.i.d. draws of } \boldsymbol{x}_n \text{ given } \boldsymbol{z}_n\text{)} \end{aligned}$$

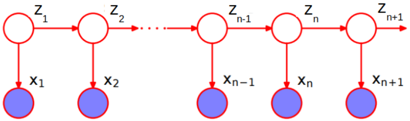


- $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is called state-transition model, $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is called observation/emission model

Latent Variable Models for Sequential Data

- For both cases (discrete/continuous $\boldsymbol{z}_n$), the generic model can be written as follows

$$
\begin{aligned}
\boldsymbol{z}_n|\boldsymbol{z}_{n-1} &\sim p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) \qquad \text{(first-order dependence b/w } \boldsymbol{z}_n\text{'s)} \\
\boldsymbol{x}_n|\boldsymbol{z}_n &\sim p(\boldsymbol{x}_n|\boldsymbol{z}_n) \qquad \text{(i.i.d. draws of } \boldsymbol{x}_n \text{ given } \boldsymbol{z}_n\text{)}
\end{aligned}
$$

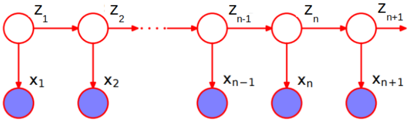


- $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is called state-transition model, $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is called observation/emission model
  - Note: In some cases, the parameters defining these distributions may be known

Latent Variable Models for Sequential Data

- For both cases (discrete/continuous $\boldsymbol{z}_n$), the generic model can be written as follows

$$\begin{aligned} \boldsymbol{z}_n|\boldsymbol{z}_{n-1} &\sim p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) && \text{(first-order dependence b/w } \boldsymbol{z}_n\text{'s)} \\ \boldsymbol{x}_n|\boldsymbol{z}_n &\sim p(\boldsymbol{x}_n|\boldsymbol{z}_n) && \text{(i.i.d. draws of } \boldsymbol{x}_n \text{ given } \boldsymbol{z}_n\text{)} \end{aligned}$$

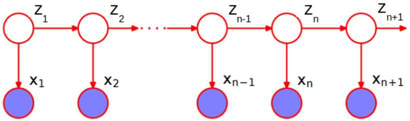


- $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is called state-transition model, $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is called observation/emission model
  - Note: In some cases, the parameters defining these distributions may be known
- If states $\boldsymbol{z}_n$ are discrete, we get a Hidden Markov Model (HMM)

Latent Variable Models for Sequential Data

- For both cases (discrete/continuous $\boldsymbol{z}_n$), the generic model can be written as follows

$$
\begin{aligned}
\boldsymbol{z}_n|\boldsymbol{z}_{n-1} &\sim p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) && \text{(first-order dependence b/w } \boldsymbol{z}_n\text{'s)} \\
\boldsymbol{x}_n|\boldsymbol{z}_n &\sim p(\boldsymbol{x}_n|\boldsymbol{z}_n) && \text{(i.i.d. draws of } \boldsymbol{x}_n \text{ given } \boldsymbol{z}_n\text{)}
\end{aligned}
$$

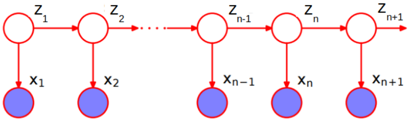


- $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is called state-transition model, $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is called observation/emission model
  - Note: In some cases, the parameters defining these distributions may be known
- If states $\boldsymbol{z}_n$ are discrete, we get a Hidden Markov Model (HMM)
- If states $\boldsymbol{z}_n$ are continuous vectors, we get a State-Space Model (SSM)

Latent Variable Models for Sequential Data

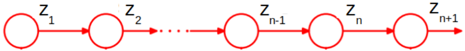
- For both cases (discrete/continuous $\boldsymbol{z}_n$), the generic model can be written as follows

$$\begin{aligned} \boldsymbol{z}_n|\boldsymbol{z}_{n-1} &\sim p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) && \text{(first-order dependence b/w } \boldsymbol{z}_n\text{'s)} \\ \boldsymbol{x}_n|\boldsymbol{z}_n &\sim p(\boldsymbol{x}_n|\boldsymbol{z}_n) && \text{(i.i.d. draws of } \boldsymbol{x}_n \text{ given } \boldsymbol{z}_n\text{)} \end{aligned}$$



- $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is called state-transition model, $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is called observation/emission model
  - Note: In some cases, the parameters defining these distributions may be known
- If states $\boldsymbol{z}_n$ are discrete, we get a Hidden Markov Model (HMM)
- If states $\boldsymbol{z}_n$ are continuous vectors, we get a State-Space Model (SSM)
- In both cases, observations $\boldsymbol{x}_n$ can be anything (discrete/real)

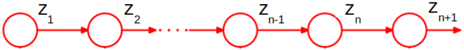
State-Transition Model



- For discrete states case (HMM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a discrete distribution, e.g.,

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$$

State-Transition Model

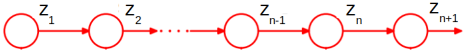


- For discrete states case (HMM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a discrete distribution, e.g.,

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$$

where $\boldsymbol{\pi}_\ell = [\pi_{\ell,1}, \ldots, \pi_{\ell,K}]$ is $K \times 1$ a transition prob. vector, s.t. $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = \ell) = \pi_{\ell,k}$

State-Transition Model

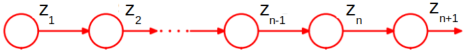


- For discrete states case (HMM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a discrete distribution, e.g.,

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$$

  where $\boldsymbol{\pi}_\ell = [\pi_{\ell,1}, \ldots, \pi_{\ell,K}]$ is $K \times 1$ a transition prob. vector, s.t. $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = \ell) = \pi_{\ell,k}$
- For HMM, $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is fully defined by a $K \times K$ transition prob. matrix $\Pi = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_K]$

State-Transition Model



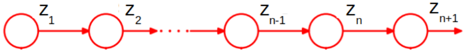
- For discrete states case (HMM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a discrete distribution, e.g.,

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$$

  where $\boldsymbol{\pi}_\ell = [\pi_{\ell,1}, \ldots, \pi_{\ell,K}]$ is $K \times 1$ a transition prob. vector, s.t. $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = \ell) = \pi_{\ell,k}$
- For HMM, $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is fully defined by a $K \times K$ transition prob. matrix $\Pi = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_K]$
- For continuous states (SSM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a continuous distribution, e.g., Gaussian

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = \mathcal{N}(\mathbf{A}\boldsymbol{z}_{n-1}, \mathbf{I}_K)$$

State-Transition Model



- For discrete states case (HMM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a discrete distribution, e.g.,

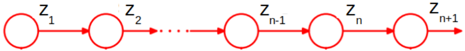
$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$$

  where $\boldsymbol{\pi}_\ell = [\pi_{\ell,1}, \ldots, \pi_{\ell,K}]$ is $K \times 1$ a transition prob. vector, s.t. $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = \ell) = \pi_{\ell,k}$
- For HMM, $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is fully defined by a $K \times K$ transition prob. matrix $\Pi = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_K]$
- For continuous states (SSM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a continuous distribution, e.g., Gaussian

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = \mathcal{N}(\mathbf{A}\boldsymbol{z}_{n-1}, \mathbf{I}_K)$$

- Note: More powerful transition models usually employ nonlinear mappings between $\boldsymbol{z}_{n-1}$ and $\boldsymbol{z}_n$

State-Transition Model



- For discrete states case (HMM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a discrete distribution, e.g.,

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$$

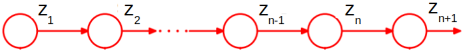
where $\boldsymbol{\pi}_\ell = [\pi_{\ell,1}, \ldots, \pi_{\ell,K}]$ is $K \times 1$ a transition prob. vector, s.t. $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = \ell) = \pi_{\ell,k}$

- For HMM, $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is fully defined by a $K \times K$ transition prob. matrix $\Pi = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_K]$
- For continuous states (SSM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a continuous distribution, e.g., Gaussian

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = \mathcal{N}(\mathbf{A}\boldsymbol{z}_{n-1}, \mathbf{I}_K)$$

- Note: More powerful transition models usually employ nonlinear mappings between $\boldsymbol{z}_{n-1}$ and $\boldsymbol{z}_n$
- For both HMM and SSM, there is also an initial state distribution $p(\boldsymbol{z}_1)$

State-Transition Model



- For discrete states case (HMM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a discrete distribution, e.g.,

$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}=\ell) = \text{multinoulli}(\boldsymbol{\pi}_\ell)$$

where $\boldsymbol{\pi}_\ell = [\pi_{\ell,1},\ldots,\pi_{\ell,K}]$ is $K\times 1$ a transition prob. vector, s.t. $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1}=\ell) = \pi_{\ell,k}$

- For HMM, $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ is fully defined by a $K\times K$ transition prob. matrix $\Pi = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_K]$
- For continuous states (SSM), $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$ will be a continuous distribution, e.g., Gaussian

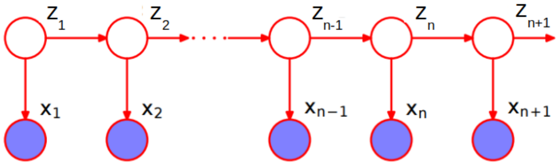
$$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = \mathcal{N}(\mathbf{A}\boldsymbol{z}_{n-1}, \mathbf{I}_K)$$

- Note: More powerful transition models usually employ nonlinear mappings between $\boldsymbol{z}_{n-1}$ and $\boldsymbol{z}_n$
- For both HMM and SSM, there is also an initial state distribution $p(\boldsymbol{z}_1)$, e.g.,

$$\begin{aligned} p(\boldsymbol{z}_1) &= \text{multinoulli}(\boldsymbol{\pi}_0) && \text{(for HMM)} \\ p(\boldsymbol{z}_1) &= \mathcal{N}(\mathbf{0}, \mathbf{I_K}) && \text{(for SSM)} \end{aligned}$$

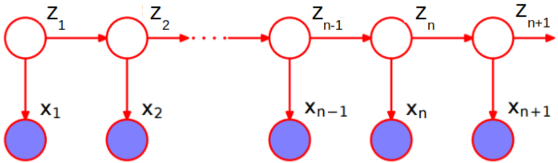
Observation/Emission Model

- The type of observation model distribution $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ depends on the type of data



Observation/Emission Model

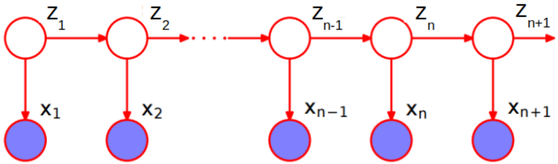
- The type of observation model distribution $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ depends on the type of data



- For discrete observations (e.g., words), $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is a discrete distribution (e.g., multinoulli)

Observation/Emission Model

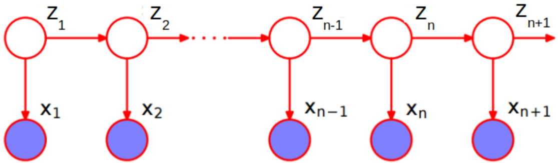
- The type of observation model distribution $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ depends on the type of data



- For discrete observations (e.g., words), $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is a discrete distribution (e.g., multinoulli)
- For continuous observations (e.g., images, location of an object, etc.), $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is a continuous distribution (e.g., Gaussian)

Observation/Emission Model

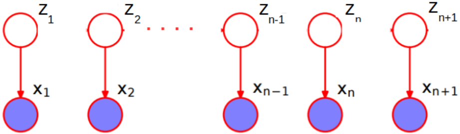
- The type of observation model distribution $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ depends on the type of data



- For discrete observations (e.g., words), $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is a discrete distribution (e.g., multinoulli)
- For continuous observations (e.g., images, location of an object, etc.), $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ is a continuous distribution (e.g., Gaussian)
- Note: More powerful observation models usually employ nonlinear mappings between $\boldsymbol{z}_n$ and $\boldsymbol{x}_n$

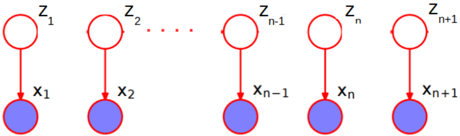
A Special Case

- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



A Special Case

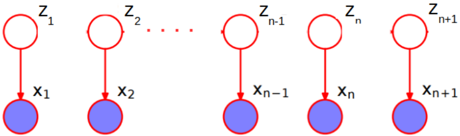
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes

A Special Case

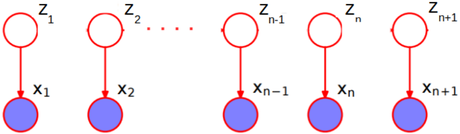
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model

A Special Case

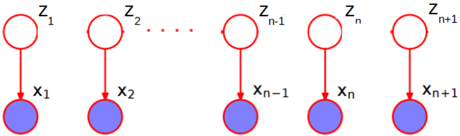
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$

A Special Case

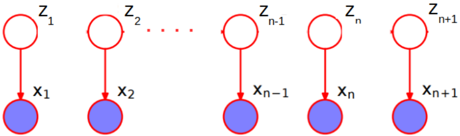
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- SSM becomes

A Special Case

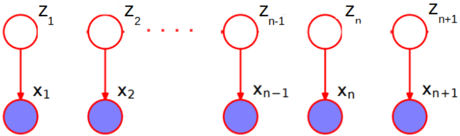
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- SSM becomes PPCA/GPLVM/DLGM

A Special Case

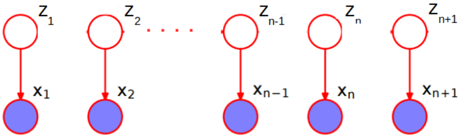
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- SSM becomes PPCA/GPLVM/DLGM. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ or $\mathcal{N}(\boldsymbol{\mu}, \Psi)$

A Special Case

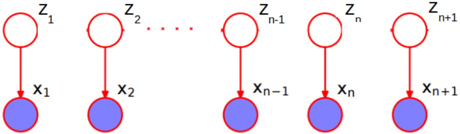
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- SSM becomes PPCA/GPLVM/DLGM. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I_K})$ or $\mathcal{N}(\boldsymbol{\mu}, \Psi)$
- Therefore, inference algorithms for HMM/SSM are often very similar to mixture models/PPCA

A Special Case

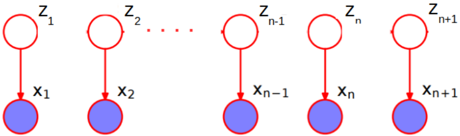
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- SSM becomes PPCA/GPLVM/DLGM. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I_K})$ or $\mathcal{N}(\boldsymbol{\mu}, \Psi)$
- Therefore, inference algorithms for HMM/SSM are often very similar to mixture models/PPCA
  - Only main difference is how the latent variables $\boldsymbol{z}_n$'s are inferred (because these are no longer i.i.d.)

A Special Case

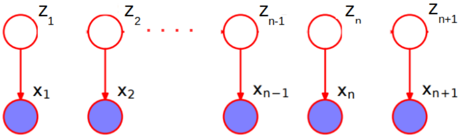
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- SSM becomes PPCA/GPLVM/DLGM. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ or $\mathcal{N}(\boldsymbol{\mu}, \Psi)$
- Therefore, inference algorithms for HMM/SSM are often very similar to mixture models/PPCA
  - Only main difference is how the latent variables $\boldsymbol{z}_n$'s are inferred (because these are no longer i.i.d.)
  - E.g., if using EM, only E step needs to change. Given the expectations, the M step updates are derived similarly to how it's done in mixture models and PPCA

A Special Case

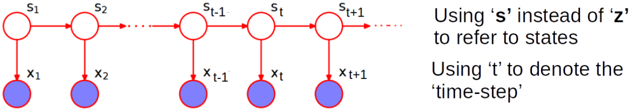
- What if we have i.i.d. latent states, i.e.,. $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n)$?



- HMM becomes a standard Mixture Model. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1} = \ell) = p(\boldsymbol{z}_n) = \text{multinoulli}(\boldsymbol{\pi})$
- SSM becomes PPCA/GPLVM/DLGM. Reason: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}) = p(\boldsymbol{z}_n) = \mathcal{N}(\mathbf{0}, \mathbf{I_K})$ or $\mathcal{N}(\boldsymbol{\mu}, \Psi)$
- Therefore, inference algorithms for HMM/SSM are often very similar to mixture models/PPCA
  - Only main difference is how the latent variables $\boldsymbol{z}_n$'s are inferred (because these are no longer i.i.d.)
  - E.g., if using EM, only E step needs to change. Given the expectations, the M step updates are derived similarly to how it's done in mixture models and PPCA (Bishop Chap 13 has EM for HMM and SSM)

State Space Models (SSM)

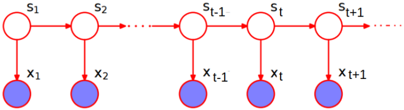
- Today we will mainly focus on SSM (when the latent variables are continuous vectors)



- Most of the details of methods we will see apply to HMMs too (but $\boldsymbol{s}_t$ will be discrete)

State Space Models (SSM)

- Today we will mainly focus on SSM (when the latent variables are continuous vectors)



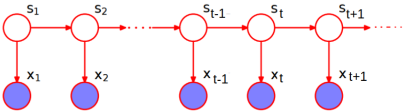
Using ‘**s**’ instead of ‘**z**’ to refer to states

Using ‘t’ to denote the ‘time-step’

- Most of the details of methods we will see apply to HMMs too (but $\boldsymbol{s}_t$ will be discrete)
- In the most general form, the transition and observation models in an SSM can be expressed as

State Space Models (SSM)

- Today we will mainly focus on SSM (when the latent variables are continuous vectors)



Using '**s**' instead of '**z**' to refer to states

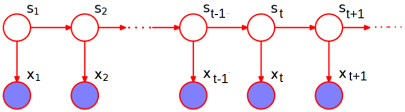
Using 't' to denote the 'time-step'

- Most of the details of methods we will see apply to HMMs too (but $\boldsymbol{s}_t$ will be discrete)
- In the most general form, the transition and observation models in an SSM can be expressed as

$$\boldsymbol{s}_t|\boldsymbol{s}_{t-1} \quad = \quad g_t(\boldsymbol{s}_{t-1}) + \epsilon_t \qquad \text{(must be a cont. dist. over } \boldsymbol{s}_t\text{)}$$

State Space Models (SSM)

- Today we will mainly focus on SSM (when the latent variables are continuous vectors)



Using '**s**' instead of '**z**' to refer to states

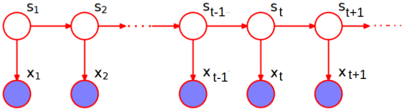
Using 't' to denote the 'time-step'

- Most of the details of methods we will see apply to HMMs too (but $\boldsymbol{s}_t$ will be discrete)
- In the most general form, the transition and observation models in an SSM can be expressed as

$$\begin{aligned} \boldsymbol{s}_t|\boldsymbol{s}_{t-1} &= g_t(\boldsymbol{s}_{t-1}) + \epsilon_t \qquad &(\text{must be a cont. dist. over } \boldsymbol{s}_t) \\ \boldsymbol{x}_t|\boldsymbol{s}_t &= h_t(\boldsymbol{s}_t) + \delta_t \qquad &(\text{can be any dist. over } \boldsymbol{x}_t) \end{aligned}$$

State Space Models (SSM)

- Today we will mainly focus on SSM (when the latent variables are continuous vectors)



Using '**s**' instead of '**z**' to refer to states

Using 't' to denote the 'time-step'

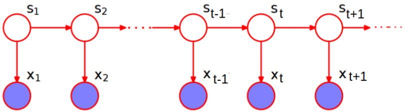
- Most of the details of methods we will see apply to HMMs too (but $\boldsymbol{s}_t$ will be discrete)
- In the most general form, the transition and observation models in an SSM can be expressed as

$$
\begin{aligned}
\boldsymbol{s}_t|\boldsymbol{s}_{t-1} &= g_t(\boldsymbol{s}_{t-1}) + \epsilon_t && \text{(must be a cont. dist. over } \boldsymbol{s}_t\text{)} \\
\boldsymbol{x}_t|\boldsymbol{s}_t &= h_t(\boldsymbol{s}_t) + \delta_t && \text{(can be any dist. over } \boldsymbol{x}_t\text{)}
\end{aligned}
$$

- Here g_t and h_t are functions (can be linear/nonlinear)

State Space Models (SSM)

- Today we will mainly focus on SSM (when the latent variables are continuous vectors)



Using '**s**' instead of '**z**' to refer to states

Using 't' to denote the 'time-step'

- Most of the details of methods we will see apply to HMMs too (but $\boldsymbol{s}_t$ will be discrete)
- In the most general form, the transition and observation models in an SSM can be expressed as

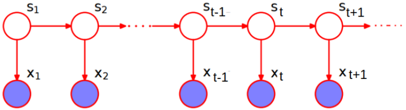
$$\begin{aligned} \boldsymbol{s}_t|\boldsymbol{s}_{t-1} &= g_t(\boldsymbol{s}_{t-1}) + \epsilon_t && \text{(must be a cont. dist. over } \boldsymbol{s}_t\text{)} \\ \boldsymbol{x}_t|\boldsymbol{s}_t &= h_t(\boldsymbol{s}_t) + \delta_t && \text{(can be any dist. over } \boldsymbol{x}_t\text{)} \end{aligned}$$

- Here g_t and h_t are functions (can be linear/nonlinear)
- Assuming zero-mean Gaussian noise $\epsilon_t \sim \mathcal{N}(0, \mathbf{Q}_t)$, $\delta_t \sim \mathcal{N}(0, \mathbf{R}_t)$, we get a Gaussian SSM

$$\begin{aligned} \boldsymbol{s}_t|\boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t|g_t(\boldsymbol{s}_{t-1}), \mathbf{Q}_t) \\ \boldsymbol{x}_t|\boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t|h_t(\boldsymbol{s}_t), \mathbf{R}_t) \end{aligned}$$

State Space Models (SSM)

- Today we will mainly focus on SSM (when the latent variables are continuous vectors)



Using '**s**' instead of '**z**' to refer to states

Using 't' to denote the 'time-step'

- Most of the details of methods we will see apply to HMMs too (but $\boldsymbol{s}_t$ will be discrete)
- In the most general form, the transition and observation models in an SSM can be expressed as

$$\begin{aligned} \boldsymbol{s}_t|\boldsymbol{s}_{t-1} &= g_t(\boldsymbol{s}_{t-1}) + \epsilon_t && \text{(must be a cont. dist. over } \boldsymbol{s}_t) \\ \boldsymbol{x}_t|\boldsymbol{s}_t &= h_t(\boldsymbol{s}_t) + \delta_t && \text{(can be any dist. over } \boldsymbol{x}_t) \end{aligned}$$

- Here g_t and h_t are functions (can be linear/nonlinear)
- Assuming zero-mean Gaussian noise $\epsilon_t \sim \mathcal{N}(0, \mathbf{Q}_t)$, $\delta_t \sim \mathcal{N}(0, \mathbf{R}_t)$, we get a Gaussian SSM

$$\begin{aligned} \boldsymbol{s}_t|\boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t|g_t(\boldsymbol{s}_{t-1}), \mathbf{Q}_t) \\ \boldsymbol{x}_t|\boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t|h_t(\boldsymbol{s}_t), \mathbf{R}_t) \end{aligned}$$

- Note: If $g_t, h_t, \mathbf{Q}_t, \mathbf{R}_t$ are independent of t then the model is called stationary

State Space Models (SSM)

- A simple example of a state-space model

$$
\begin{aligned}
\boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= \boldsymbol{s}_{t-1} + \epsilon_t \\
\boldsymbol{x}_t | \boldsymbol{s}_t &= \boldsymbol{s}_t + \delta_t \qquad \text{(assumes } \boldsymbol{x}_t \text{ and } \boldsymbol{s}_t \text{ to be of same size)}
\end{aligned}
$$

State Space Models (SSM)

- A simple example of a state-space model

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= \boldsymbol{s}_{t-1} + \epsilon_t \\ \boldsymbol{x}_t | \boldsymbol{s}_t &= \boldsymbol{s}_t + \delta_t \qquad \text{(assumes } \boldsymbol{x}_t \text{ and } \boldsymbol{s}_t \text{ to be of same size)} \end{aligned}$$

- Another simple but more general example (latent states and observations of diff. dimensions)

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= \mathbf{A}_t \boldsymbol{s}_{t-1} + \epsilon_t \qquad (\mathbf{A}_t \text{ is } K \times K) \\ \boldsymbol{x}_t | \boldsymbol{s}_t &= \mathbf{B}_t \boldsymbol{s}_t + \delta_t \qquad (\mathbf{B}_t \text{ is } D \times K) \end{aligned}$$

State Space Models (SSM)

- A simple example of a state-space model

$$\begin{aligned} \boldsymbol{s}_t|\boldsymbol{s}_{t-1} &= \boldsymbol{s}_{t-1} + \epsilon_t \\ \boldsymbol{x}_t|\boldsymbol{s}_t &= \boldsymbol{s}_t + \delta_t \end{aligned}$$

(assumes $\boldsymbol{x}_t$ and $\boldsymbol{s}_t$ to be of same size)

- Another simple but more general example (latent states and observations of diff. dimensions)

$$\begin{aligned} \boldsymbol{s}_t|\boldsymbol{s}_{t-1} &= \mathbf{A}_t\boldsymbol{s}_{t-1} + \epsilon_t && (\mathbf{A}_t \text{ is } K \times K) \\ \boldsymbol{x}_t|\boldsymbol{s}_t &= \mathbf{B}_t\boldsymbol{s}_t + \delta_t && (\mathbf{B}_t \text{ is } D \times K) \end{aligned}$$

- The above can also be written as follows

$$\begin{aligned} \boldsymbol{s}_t|\boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}_t\boldsymbol{s}_{t-1}, \mathbf{Q}_t) \\ \boldsymbol{x}_t|\boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}_t\boldsymbol{s}_t, \mathbf{R}_t) \end{aligned}$$

State Space Models (SSM)

- A simple example of a state-space model

$$
\begin{aligned}
\boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= \boldsymbol{s}_{t-1} + \epsilon_t \\
\boldsymbol{x}_t | \boldsymbol{s}_t &= \boldsymbol{s}_t + \delta_t \qquad \text{(assumes } \boldsymbol{x}_t \text{ and } \boldsymbol{s}_t \text{ to be of same size)}
\end{aligned}
$$

- Another simple but more general example (latent states and observations of diff. dimensions)

$$
\begin{aligned}
\boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= \mathbf{A}_t \boldsymbol{s}_{t-1} + \epsilon_t \qquad (\mathbf{A}_t \text{ is } K \times K) \\
\boldsymbol{x}_t | \boldsymbol{s}_t &= \mathbf{B}_t \boldsymbol{s}_t + \delta_t \qquad (\mathbf{B}_t \text{ is } D \times K)
\end{aligned}
$$

- The above can also be written as follows

$$
\begin{aligned}
\boldsymbol{s}_t | \boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t | \mathbf{A}_t \boldsymbol{s}_{t-1}, \mathbf{Q}_t) \\
\boldsymbol{x}_t | \boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t | \mathbf{B}_t \boldsymbol{s}_t, \mathbf{R}_t)
\end{aligned}
$$

- This is a Linear Gaussian SSM; also called Linear Dynamical System (LDS)

State Space Models (SSM)

- A simple example of a state-space model

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= \boldsymbol{s}_{t-1} + \epsilon_t \\ \boldsymbol{x}_t | \boldsymbol{s}_t &= \boldsymbol{s}_t + \delta_t \qquad \text{(assumes } \boldsymbol{x}_t \text{ and } \boldsymbol{s}_t \text{ to be of same size)} \end{aligned}$$

- Another simple but more general example (latent states and observations of diff. dimensions)

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= \mathbf{A}_t \boldsymbol{s}_{t-1} + \epsilon_t \qquad (\mathbf{A}_t \text{ is } K \times K) \\ \boldsymbol{x}_t | \boldsymbol{s}_t &= \mathbf{B}_t \boldsymbol{s}_t + \delta_t \qquad (\mathbf{B}_t \text{ is } D \times K) \end{aligned}$$

- The above can also be written as follows

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t | \mathbf{A}_t \boldsymbol{s}_{t-1}, \mathbf{Q}_t) \\ \boldsymbol{x}_t | \boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t | \mathbf{B}_t \boldsymbol{s}_t, \mathbf{R}_t) \end{aligned}$$

- This is a Linear Gaussian SSM; also called Linear Dynamical System (LDS)
- Note: $\mathbf{A}_t$, $\mathbf{B}_t, \mathbf{Q}_t, \mathbf{R}_t$ may be known (fixed) or may be required to be learned

Linear Gaussian SSM (LDS): An Example

- Consider the linear Gaussian SSM: $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} = \mathbf{A}_t\boldsymbol{s}_{t-1} + \epsilon_t$ and $\boldsymbol{x}_t|\boldsymbol{s}_t = \mathbf{B}_t\boldsymbol{s}_t + \delta_t$

Linear Gaussian SSM (LDS): An Example

- Consider the linear Gaussian SSM: $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} = \mathbf{A}_t\boldsymbol{s}_{t-1} + \epsilon_t$ and $\boldsymbol{x}_t|\boldsymbol{s}_t = \mathbf{B}_t\boldsymbol{s}_t + \delta_t$
- Suppose $\boldsymbol{x}_t \in \mathbb{R}^2$ denotes the (noisy) observed 2D location of an object

Linear Gaussian SSM (LDS): An Example

- Consider the linear Gaussian SSM: $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} = \mathbf{A}_t\boldsymbol{s}_{t-1} + \epsilon_t$ and $\boldsymbol{x}_t|\boldsymbol{s}_t = \mathbf{B}_t\boldsymbol{s}_t + \delta_t$
- Suppose $\boldsymbol{x}_t \in \mathbb{R}^2$ denotes the (noisy) observed 2D location of an object
- Suppose $\boldsymbol{s}_t \in \mathbb{R}^6$ denotes its "state" vector $\boldsymbol{s}_t = [pos_1, vel_1, accel_1, pos_2, vel_2, accel_2]$

Linear Gaussian SSM (LDS): An Example

- Consider the linear Gaussian SSM: $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} = \mathbf{A}_t\boldsymbol{s}_{t-1} + \epsilon_t$ and $\boldsymbol{x}_t|\boldsymbol{s}_t = \mathbf{B}_t\boldsymbol{s}_t + \delta_t$
- Suppose $\boldsymbol{x}_t \in \mathbb{R}^2$ denotes the (noisy) observed 2D location of an object
- Suppose $\boldsymbol{s}_t \in \mathbb{R}^6$ denotes its "state" vector $\boldsymbol{s}_t = [pos_1, vel_1, accel_1, pos_2, vel_2, accel_2]$
- Assuming a pre-defined $\mathbf{A}_t$, $\mathbf{B}_t$, a possible linear Gaussian SSM to model this data will be

$$\mathbf{s}_t = \underbrace{\boxed{\begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & e^{-\alpha\Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & e^{-\alpha\Delta t} \end{bmatrix}}}_{\mathbf{A}_t} \mathbf{s}_{t-1} + \epsilon_t$$

Linear Gaussian SSM (LDS): An Example

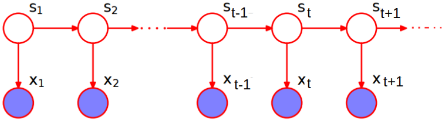
- Consider the linear Gaussian SSM: $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} = \mathbf{A}_t\boldsymbol{s}_{t-1} + \epsilon_t$ and $\boldsymbol{x}_t|\boldsymbol{s}_t = \mathbf{B}_t\boldsymbol{s}_t + \delta_t$
- Suppose $\boldsymbol{x}_t \in \mathbb{R}^2$ denotes the (noisy) observed 2D location of an object
- Suppose $\boldsymbol{s}_t \in \mathbb{R}^6$ denotes its "state" vector $\boldsymbol{s}_t = [pos_1, vel_1, accel_1, pos_2, vel_2, accel_2]$
- Assuming a pre-defined $\mathbf{A}_t$, $\mathbf{B}_t$, a possible linear Gaussian SSM to model this data will be

$$\mathbf{s}_t = \underbrace{\boxed{\begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & e^{-\alpha\Delta t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & e^{-\alpha\Delta t} \end{bmatrix}}}_{\mathbf{A}_t} \mathbf{s}_{t-1} + \epsilon_t$$

$$\mathbf{x}_t = \underbrace{\boxed{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}}_{\mathbf{B}_t} \mathbf{s}_t + \delta_t$$

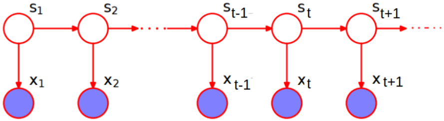
Typical Inference Tasks in Gaussian SSM

- One of the key tasks: Given sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots,$ infer the latent states $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots$



Typical Inference Tasks in Gaussian SSM

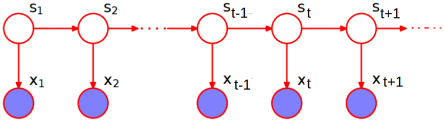
- One of the key tasks: Given sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$, infer the latent states $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots$



- This is usually solves in one of the following two ways

Typical Inference Tasks in Gaussian SSM

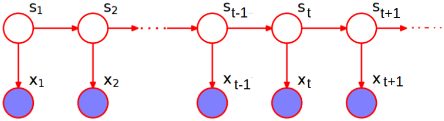
- One of the key tasks: Given sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$, infer the latent states $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots$



- This is usually solves in one of the following two ways
  - Infer the distribution $p(\boldsymbol{s}_t | \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ given the past observations: "Filtering Problem"

Typical Inference Tasks in Gaussian SSM

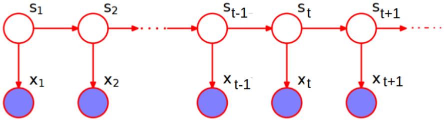
- One of the key tasks: Given sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$, infer the latent states $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots$



- This is usually solves in one of the following two ways
  - Infer the distribution $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ given the past observations: "Filtering Problem"
  - Infer the distribution $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given all (past/future) observations: "Smoothing Problem"

Typical Inference Tasks in Gaussian SSM

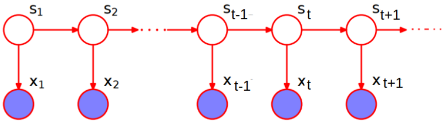
- One of the key tasks: Given sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$, infer the latent states $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots$



- This is usually solves in one of the following two ways
  - Infer the distribution $p(\boldsymbol{s}_t | \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ given the past observations: "Filtering Problem"
  - Infer the distribution $p(\boldsymbol{s}_t | \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given all (past/future) observations: "Smoothing Problem"
- Other tasks we may be interested in

Typical Inference Tasks in Gaussian SSM

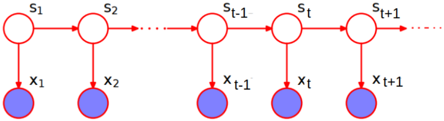
- One of the key tasks: Given sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$, infer the latent states $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots$



- This is usually solves in one of the following two ways
  - Infer the distribution $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ given the past observations: "Filtering Problem"
  - Infer the distribution $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given all (past/future) observations: "Smoothing Problem"
- Other tasks we may be interested in
  - Predicting future state(s) given observations seen thus far: $p(\boldsymbol{s}_{t+h}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ for $h \geq 1$

Typical Inference Tasks in Gaussian SSM

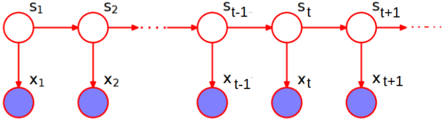
- One of the key tasks: Given sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$, infer the latent states $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots$



- This is usually solves in one of the following two ways
  - Infer the distribution $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ given the past observations: "Filtering Problem"
  - Infer the distribution $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given all (past/future) observations: "Smoothing Problem"
- Other tasks we may be interested in
  - Predicting future state(s) given observations seen thus far: $p(\boldsymbol{s}_{t+h}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ for $h \geq 1$
  - Predict next observation(s) given observations seen thus far: $p(\boldsymbol{x}_{t+h}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ for $h \geq 1$

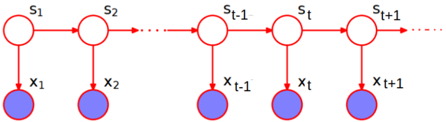
Typical Inference Tasks in Gaussian SSM

- One of the key tasks: Given sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots$, infer the latent states $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \ldots$



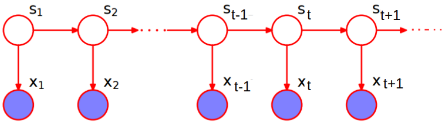
- This is usually solves in one of the following two ways
  - Infer the distribution $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ given the past observations: "Filtering Problem"
  - Infer the distribution $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given all (past/future) observations: "Smoothing Problem"
- Other tasks we may be interested in
  - Predicting future state(s) given observations seen thus far: $p(\boldsymbol{s}_{t+h}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ for $h \geq 1$
  - Predict next observation(s) given observations seen thus far: $p(\boldsymbol{x}_{t+h}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$ for $h \geq 1$
- Today, we'll mainly focus on the filtering problem (solved using the Kalman Filtering algorithm)

Kalman Filtering



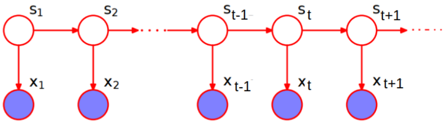
- Recall that $\boldsymbol{s}_t | \boldsymbol{s}_{t-1} \sim \mathcal{N}(\boldsymbol{s}_t | \mathbf{A}_t \boldsymbol{s}_{t-1}, \mathbf{Q}_t)$ and $\boldsymbol{x}_t | \boldsymbol{s}_t \sim \mathcal{N}(\boldsymbol{x}_t | \mathbf{B}_t \boldsymbol{s}_t, \mathbf{R}_t)$

Kalman Filtering



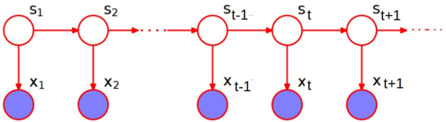
- Recall that $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} \sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}_t\boldsymbol{s}_{t-1}, \mathbf{Q}_t)$ and $\boldsymbol{x}_t|\boldsymbol{s}_t \sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}_t\boldsymbol{s}_t, \mathbf{R}_t)$
- Let's assume a stationary SSM, i.e., $\mathbf{A}_t = \mathbf{A}$, $\mathbf{B}_t = \mathbf{B}$, $\mathbf{Q}_t = \mathbf{Q}$, and $\mathbf{R}_t = \mathbf{R}$

Kalman Filtering



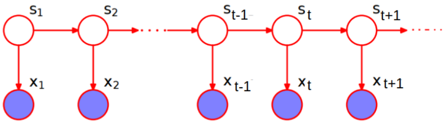
- Recall that $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} \sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}_t\boldsymbol{s}_{t-1}, \mathbf{Q}_t)$ and $\boldsymbol{x}_t|\boldsymbol{s}_t \sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}_t\boldsymbol{s}_t, \mathbf{R}_t)$
- Let's assume a stationary SSM, i.e., $\mathbf{A}_t = \mathbf{A}$, $\mathbf{B}_t = \mathbf{B}$, $\mathbf{Q}_t = \mathbf{Q}$, and $\mathbf{R}_t = \mathbf{R}$
- Kalman Filtering gives an exact way to infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ in a linear Gaussian SSM

Kalman Filtering



- Recall that $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} \sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}_t\boldsymbol{s}_{t-1}, \mathbf{Q}_t)$ and $\boldsymbol{x}_t|\boldsymbol{s}_t \sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}_t\boldsymbol{s}_t, \mathbf{R}_t)$
- Let's assume a stationary SSM, i.e., $\mathbf{A}_t = \mathbf{A}$, $\mathbf{B}_t = \mathbf{B}$, $\mathbf{Q}_t = \mathbf{Q}$, and $\mathbf{R}_t = \mathbf{R}$
- Kalman Filtering gives an exact way to infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ in a linear Gaussian SSM
  - Note: The "exactness" assumes we are given $\mathbf{A}$, $\mathbf{B}, \mathbf{Q}, \mathbf{R}$ are known (or have estimated these)

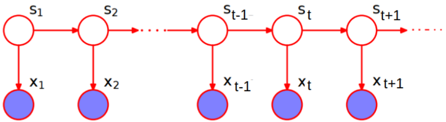
Kalman Filtering



- Recall that $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} \sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}_t\boldsymbol{s}_{t-1}, \mathbf{Q}_t)$ and $\boldsymbol{x}_t|\boldsymbol{s}_t \sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}_t\boldsymbol{s}_t, \mathbf{R}_t)$
- Let's assume a stationary SSM, i.e., $\mathbf{A}_t = \mathbf{A}$, $\mathbf{B}_t = \mathbf{B}$, $\mathbf{Q}_t = \mathbf{Q}$, and $\mathbf{R}_t = \mathbf{R}$
- Kalman Filtering gives an exact way to infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ in a linear Gaussian SSM
  - Note: The "exactness" assumes we are given $\mathbf{A}$, $\mathbf{B}, \mathbf{Q}, \mathbf{R}$ are known (or have estimated these)
- Using Bayes rule, our target will be

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \propto p(\boldsymbol{x}_t|\boldsymbol{s}_t)\color{red}{p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1})}$$

Kalman Filtering



- Recall that $\boldsymbol{s}_t|\boldsymbol{s}_{t-1} \sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}_t\boldsymbol{s}_{t-1}, \mathbf{Q}_t)$ and $\boldsymbol{x}_t|\boldsymbol{s}_t \sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}_t\boldsymbol{s}_t, \mathbf{R}_t)$
- Let's assume a stationary SSM, i.e., $\mathbf{A}_t = \mathbf{A}$, $\mathbf{B}_t = \mathbf{B}$, $\mathbf{Q}_t = \mathbf{Q}$, and $\mathbf{R}_t = \mathbf{R}$
- Kalman Filtering gives an exact way to infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ in a linear Gaussian SSM
  - Note: The "exactness" assumes we are given $\mathbf{A}$, $\mathbf{B}, \mathbf{Q}, \mathbf{R}$ are known (or have estimated these)
- Using Bayes rule, our target will be

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \propto p(\boldsymbol{x}_t|\boldsymbol{s}_t)\color{red}{p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1})}$$

- The "prior" above is: $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1}) = \int p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1})d\boldsymbol{s}_{t-1}$

Kalman Filtering

- Thus the Kalman Filtering problem computes the following

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_t) \propto \underbrace{p(\boldsymbol{x}_t|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t,\mathbf{R})} \int \underbrace{p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})}_{\mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{s}_{t-1},\mathbf{Q})} p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_{t-1})d\boldsymbol{s}_{t-1}$$

Kalman Filtering

- Thus the Kalman Filtering problem computes the following

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_t) \propto \underbrace{p(\boldsymbol{x}_t|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t,\mathbf{R})} \int \underbrace{p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})}_{\mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{s}_{t-1},\mathbf{Q})} p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_{t-1})d\boldsymbol{s}_{t-1}$$

- Note that the LHS is the posterior on $\boldsymbol{s}_t$, the RHS consists of a posterior on $\boldsymbol{s}_{t-1}$

Kalman Filtering

- Thus the Kalman Filtering problem computes the following

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \propto \underbrace{p(\boldsymbol{x}_t|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t, \mathbf{R})} \int \underbrace{p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})}_{\mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{s}_{t-1}, \mathbf{Q})} p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1}) d\boldsymbol{s}_{t-1}$$

- Note that the LHS is the posterior on $\boldsymbol{s}_t$, the RHS consists of a posterior on $\boldsymbol{s}_{t-1}$
- This suggests a simple "forward algorithm" to recursively compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$

Kalman Filtering

- Thus the Kalman Filtering problem computes the following

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \propto \underbrace{p(\boldsymbol{x}_t|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t,\mathbf{R})} \int \underbrace{p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})}_{\mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{s}_{t-1},\mathbf{Q})} p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1}) d\boldsymbol{s}_{t-1}$$

- Note that the LHS is the posterior on $\boldsymbol{s}_t$, the RHS consists of a posterior on $\boldsymbol{s}_{t-1}$
- This suggests a simple "forward algorithm" to recursively compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$
  - For Kalman smoothing problem $p(\boldsymbol{z}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T)$, a similar recursive "forward-backward" algorithm exists (the backup slides contain an illustration for the same)

Kalman Filtering

- Thus the Kalman Filtering problem computes the following

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \propto \underbrace{p(\boldsymbol{x}_t|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t,\mathbf{R})} \int \underbrace{p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})}_{\mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{s}_{t-1},\mathbf{Q})} p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1}) d\boldsymbol{s}_{t-1}$$

- Note that the LHS is the posterior on $\boldsymbol{s}_t$, the RHS consists of a posterior on $\boldsymbol{s}_{t-1}$
- This suggests a simple "forward algorithm" to recursively compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$
  - For Kalman smoothing problem $p(\boldsymbol{z}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T)$, a similar recursive "forward-backward" algorithm exists (the backup slides contain an illustration for the same)
- In this Linear Gaussian SSM, $p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1})$ would be a Gausian, say $\mathcal{N}(\boldsymbol{s}_{t-1}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Kalman Filtering

- Thus the Kalman Filtering problem computes the following

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_t) \propto \underbrace{p(\boldsymbol{x}_t|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t,\mathbf{R})} \int \underbrace{p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})}_{\mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{s}_{t-1},\mathbf{Q})} p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_{t-1})d\boldsymbol{s}_{t-1}$$

- Note that the LHS is the posterior on $\boldsymbol{s}_t$, the RHS consists of a posterior on $\boldsymbol{s}_{t-1}$
- This suggests a simple "forward algorithm" to recursively compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_t)$
  - For Kalman smoothing problem $p(\boldsymbol{z}_t|\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{x}_T)$, a similar recursive "forward-backward" algorithm exists (the backup slides contain an illustration for the same)
- In this Linear Gaussian SSM, $p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_{t-1})$ would be a Gausian, say $\mathcal{N}(\boldsymbol{s}_{t-1}|\boldsymbol{\mu},\boldsymbol{\Sigma})$
  - Reason: Starting with $p(\boldsymbol{s}_0)=\mathcal{N}(\boldsymbol{s}_0|\mathbf{0},\mathbf{I}_K)$, the posterior over $\boldsymbol{s}_t$ will be Gaussian at each step t

Kalman Filtering

- Thus the Kalman Filtering problem computes the following

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \propto \underbrace{p(\boldsymbol{x}_t|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t, \mathbf{R})} \int \underbrace{p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1})}_{\mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{s}_{t-1}, \mathbf{Q})} p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1}) d\boldsymbol{s}_{t-1}$$

- Note that the LHS is the posterior on $\boldsymbol{s}_t$, the RHS consists of a posterior on $\boldsymbol{s}_{t-1}$
- This suggests a simple "forward algorithm" to recursively compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$
  - For Kalman smoothing problem $p(\boldsymbol{z}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_T)$, a similar recursive "forward-backward" algorithm exists (the backup slides contain an illustration for the same)
- In this Linear Gaussian SSM, $p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{t-1})$ would be a Gausian, say $\mathcal{N}(\boldsymbol{s}_{t-1}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$
  - Reason: Starting with $p(\boldsymbol{s}_0) = \mathcal{N}(\boldsymbol{s}_0|\mathbf{0}, \mathbf{I}_K)$, the posterior over $\boldsymbol{s}_t$ will be Gaussian at each step t
- Also, using Gaussian's properties, we know that

$$\int \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{s}_{t-1}, \mathbf{Q})\mathcal{N}(\boldsymbol{s}_{t-1}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) d\boldsymbol{s}_{t-1} = \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{\mu}, \mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)$$

Kalman Filtering

- We can now compute the desired posterior

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \quad \propto \quad \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t, \mathbf{R}) \times \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\mu, \mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)$$

Kalman Filtering

- We can now compute the desired posterior

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \quad \propto \quad \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t, \mathbf{R}) \times \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{\mu}, \mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)$$

- This again is a Gaussian (Gaussian likelihood and Gaussian prior), given by

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$$

Kalman Filtering

- We can now compute the desired posterior

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) \quad \propto \quad \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t, \mathbf{R}) \times \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{\mu}, \mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)$$

- This again is a Gaussian (Gaussian likelihood and Gaussian prior), given by

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$$

where the Gaussian posterior's covariance matrix and mean vector are given by

$$\begin{aligned} \boldsymbol{\Sigma}' &= [(\mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)^{-1} + \mathbf{B}^\top\mathbf{R}^{-1}\mathbf{B}]^{-1} \\ \boldsymbol{\mu}' &= \boldsymbol{\Sigma}'[\mathbf{B}^\top\mathbf{R}^{-1}\boldsymbol{x}_t + (\mathbf{Q} + \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)^{-1}\mathbf{A}\boldsymbol{\mu}] \end{aligned}$$

Kalman Filtering

- We can now compute the desired posterior

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_t) \quad \propto \quad \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}\boldsymbol{s}_t,\mathbf{R}) \times \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}\boldsymbol{\mu},\mathbf{Q}+\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)$$

- This again is a Gaussian (Gaussian likelihood and Gaussian prior), given by

$$p(\boldsymbol{s}_t|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}',\boldsymbol{\Sigma}')$$

  where the Gaussian posterior's covariance matrix and mean vector are given by

$$\begin{aligned} \boldsymbol{\Sigma}' &= [(\mathbf{Q}+\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)^{-1} + \mathbf{B}^\top\mathbf{R}^{-1}\mathbf{B}]^{-1} \\ \boldsymbol{\mu}' &= \boldsymbol{\Sigma}'[\mathbf{B}^\top\mathbf{R}^{-1}\boldsymbol{x}_t + (\mathbf{Q}+\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^\top)^{-1}\mathbf{A}\boldsymbol{\mu}] \end{aligned}$$

- Thus we get closed form expressions for the parameters $(\boldsymbol{\Sigma}',\boldsymbol{\mu}')$ of $p(\boldsymbol{s}_t|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_t)$ in terms of the parameters $(\boldsymbol{\Sigma},\boldsymbol{\mu})$ of $p(\boldsymbol{s}_{t-1}|\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_{t-1})$

Kalman Filtering: Predicting Future Observations

- We saw how to compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ which was a Gaussian $\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$

Kalman Filtering: Predicting Future Observations

- We saw how to compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ which was a Gaussian $\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$
- Often we are also interested in predicting the future observations

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)$$

Kalman Filtering: Predicting Future Observations

- We saw how to compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ which was a Gaussian $\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$
- Often we are also interested in predicting the future observations

$$p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) = \int p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1}) p(\boldsymbol{s}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) d\boldsymbol{s}_{t+1}$$

Kalman Filtering: Predicting Future Observations

- We saw how to compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ which was a Gaussian $\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$
- Often we are also interested in predicting the future <u>observations</u>

$$\begin{aligned}
p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) &= \int p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1}) \color{red}{p(\boldsymbol{s}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t)} \color{black} d\boldsymbol{s}_{t+1} \\
&= \int \underbrace{p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1})}_{\mathcal{N}(\boldsymbol{x}_{t+1}|\mathbf{B}\boldsymbol{s}_{t+1}, \mathbf{R})}
\end{aligned}$$

Kalman Filtering: Predicting Future Observations

- We saw how to compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ which was a Gaussian $\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$
- Often we are also interested in predicting the future observations

$$
\begin{aligned}
p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) &= \int p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1}) p(\boldsymbol{s}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) d\boldsymbol{s}_{t+1} \\
&= \int \underbrace{p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1})}_{\mathcal{N}(\boldsymbol{x}_{t+1}|\mathbf{B}\boldsymbol{s}_{t+1}, \mathbf{R})} \int \underbrace{p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{s}_{t+1}|\mathbf{A}\boldsymbol{s}_t, \mathbf{Q})}
\end{aligned}
$$

Kalman Filtering: Predicting Future Observations

- We saw how to compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ which was a Gaussian $\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$
- Often we are also interested in predicting the future observations

$$
\begin{aligned}
p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) &= \int p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1}) p(\boldsymbol{s}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) d\boldsymbol{s}_{t+1} \\
&= \int \underbrace{p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1})}_{\mathcal{N}(\boldsymbol{x}_{t+1}|\mathbf{B}\boldsymbol{s}_{t+1}, \mathbf{R})} \int \underbrace{p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{s}_{t+1}|\mathbf{A}\boldsymbol{s}_t, \mathbf{Q})} \underbrace{p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)}_{\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')} d\boldsymbol{s}_t d\boldsymbol{s}_{t+1}
\end{aligned}
$$

Kalman Filtering: Predicting Future Observations

- We saw how to compute $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)$ which was a Gaussian $\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')$
- Often we are also interested in predicting the future observations

$$\begin{aligned} p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) &= \int p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1}) p(\boldsymbol{s}_{t+1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t) d\boldsymbol{s}_{t+1} \\ &= \int \underbrace{p(\boldsymbol{x}_{t+1}|\boldsymbol{s}_{t+1})}_{\mathcal{N}(\boldsymbol{x}_{t+1}|\mathbf{B}\boldsymbol{s}_{t+1}, \mathbf{R})} \int \underbrace{p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t)}_{\mathcal{N}(\boldsymbol{s}_{t+1}|\mathbf{A}\boldsymbol{s}_t, \mathbf{Q})} \underbrace{p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t)}_{\mathcal{N}(\boldsymbol{s}_t|\boldsymbol{\mu}', \boldsymbol{\Sigma}')} d\boldsymbol{s}_t d\boldsymbol{s}_{t+1} \end{aligned}$$

- This requires two integrals but the final result is again a Gaussian (expression not shown here)

Kalman Filtering: Some Notes

- Note that we assumed the LDS parameters $\mathbf{A}_t$, $\mathbf{B}_t, \mathbf{Q}_t, \mathbf{R}_t$ are known

Kalman Filtering: Some Notes

- Note that we assumed the LDS parameters $\mathbf{A}_t$, $\mathbf{B}_t, \mathbf{Q}_t, \mathbf{R}_t$ are known

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t | \mathbf{A}_t \boldsymbol{s}_{t-1}, \mathbf{Q}_t) \\ \boldsymbol{x}_t | \boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t | \mathbf{B}_t \boldsymbol{s}_t, \mathbf{R}_t) \end{aligned}$$

Kalman Filtering: Some Notes

- Note that we assumed the LDS parameters $\mathbf{A}_t$, $\mathbf{B}_t, \mathbf{Q}_t, \mathbf{R}_t$ are known

$$
\begin{aligned}
\boldsymbol{s}_t|\boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}_t\boldsymbol{s}_{t-1}, \mathbf{Q}_t) \\
\boldsymbol{x}_t|\boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}_t\boldsymbol{s}_t, \mathbf{R}_t)
\end{aligned}
$$

- Usually these aren't known (unless we have some domain knowledge about the underlying system)

Kalman Filtering: Some Notes

- Note that we assumed the LDS parameters $\mathbf{A}_t$, $\mathbf{B}_t, \mathbf{Q}_t, \mathbf{R}_t$ are known

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t | \mathbf{A}_t \boldsymbol{s}_{t-1}, \mathbf{Q}_t) \\ \boldsymbol{x}_t | \boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t | \mathbf{B}_t \boldsymbol{s}_t, \mathbf{R}_t) \end{aligned}$$

- Usually these aren't known (unless we have some domain knowledge about the underlying system)
- We can use iterative methods to estimate these parameters

Kalman Filtering: Some Notes

- Note that we assumed the LDS parameters $\mathbf{A}_t$, $\mathbf{B}_t, \mathbf{Q}_t, \mathbf{R}_t$ are known

$$
\begin{aligned}
\boldsymbol{s}_t|\boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}_t\boldsymbol{s}_{t-1}, \mathbf{Q}_t) \\
\boldsymbol{x}_t|\boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}_t\boldsymbol{s}_t, \mathbf{R}_t)
\end{aligned}
$$

- Usually these aren't known (unless we have some domain knowledge about the underlying system)
- We can use iterative methods to estimate these parameters
  - Basically, we can alternate between inferring the states and inferring the parameters

Kalman Filtering: Some Notes

- Note that we assumed the LDS parameters $\mathbf{A}_t$, $\mathbf{B}_t, \mathbf{Q}_t, \mathbf{R}_t$ are known

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &\sim \mathcal{N}(\boldsymbol{s}_t | \mathbf{A}_t \boldsymbol{s}_{t-1}, \mathbf{Q}_t) \\ \boldsymbol{x}_t | \boldsymbol{s}_t &\sim \mathcal{N}(\boldsymbol{x}_t | \mathbf{B}_t \boldsymbol{s}_t, \mathbf{R}_t) \end{aligned}$$

- Usually these aren't known (unless we have some domain knowledge about the underlying system)
- We can use iterative methods to estimate these parameters
    - Basically, we can alternate between inferring the states and inferring the parameters
- This can be done using approximate inference methods such as EM, MCMC, or VB

An Application: Online Dynamic Linear Regression

- Consider a dynamic linear model for regression
- The underlying ("true") weight vector $\boldsymbol{w}$ is not static (fixed) but can change with each observation

$$\begin{aligned}
\boldsymbol{w}_t &= \boldsymbol{w}_{t-1} + \eta_t \quad \text{where} \quad \eta_t \sim \mathcal{N}(0, \mathbf{I}) \\
y_t &= \boldsymbol{x}_t^\top \boldsymbol{w}_t + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2)
\end{aligned}$$

An Application: Online Dynamic Linear Regression

- Consider a dynamic linear model for regression
- The underlying ("true") weight vector $\boldsymbol{w}$ is not static (fixed) but can change with each observation

$$\begin{aligned} \boldsymbol{w}_t &= \boldsymbol{w}_{t-1} + \eta_t \quad \text{where} \quad \eta_t \sim \mathcal{N}(0, \mathbf{I}) \\ y_t &= \boldsymbol{x}_t^\top \boldsymbol{w}_t + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2) \end{aligned}$$

- Can model it as a linear dynamical system (Kalman Filtering) problem with $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T$ as the states and $y_1, \ldots, y_T$ as observations

$$\begin{aligned} p(\boldsymbol{w}_t | \boldsymbol{w}_{t-1}) &= \mathcal{N}(\boldsymbol{w}_t | \mathbf{I}\boldsymbol{w}_{t-1}, \mathbf{I}) \qquad \text{(state-transition model)} \\ p(y_t | \boldsymbol{x}_t, \boldsymbol{w}_t) &= \mathcal{N}(y_t | \boldsymbol{x}_t^\top \boldsymbol{w}_t, \sigma^2) \qquad \text{(observation model)} \end{aligned}$$

An Application: Online Dynamic Linear Regression

- Consider a dynamic linear model for regression
- The underlying ("true") weight vector $\boldsymbol{w}$ is not static (fixed) but can change with each observation

$$
\begin{aligned}
\boldsymbol{w}_t &= \boldsymbol{w}_{t-1} + \eta_t \quad \text{where} \quad \eta_t \sim \mathcal{N}(0, \mathbf{I}) \\
y_t &= \boldsymbol{x}_t^\top \boldsymbol{w}_t + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2)
\end{aligned}
$$

- Can model it as a linear dynamical system (Kalman Filtering) problem with $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_T$ as the states and $y_1, \ldots, y_T$ as observations

$$
\begin{aligned}
p(\boldsymbol{w}_t | \boldsymbol{w}_{t-1}) &= \mathcal{N}(\boldsymbol{w}_t | \mathbf{I}\boldsymbol{w}_{t-1}, \mathbf{I}) \qquad \text{(state-transition model)} \\
p(y_t | \boldsymbol{x}_t, \boldsymbol{w}_t) &= \mathcal{N}(y_t | \boldsymbol{x}_t^\top \boldsymbol{w}_t, \sigma^2) \qquad \text{(observation model)}
\end{aligned}
$$

- Can now apply Kalman filtering to solve for $\boldsymbol{w}_t$ and y_t at future time-steps (note: this problem is also called the "recursive least squares" problem)

Other Extensions of SSM/LDS

- Nonlinear dynamical systems: Assume state-transition and observation models to be nonlinear

$$
\begin{aligned}
\boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= g(\boldsymbol{s}_{t-1}) + \epsilon_t \\
\boldsymbol{x}_t | \boldsymbol{s}_t &= h(\boldsymbol{s}_t) + \delta_t
\end{aligned}
$$

Other Extensions of SSM/LDS

- Nonlinear dynamical systems: Assume state-transition and observation models to be nonlinear

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= g(\boldsymbol{s}_{t-1}) + \epsilon_t \\ \boldsymbol{x}_t | \boldsymbol{s}_t &= h(\boldsymbol{s}_t) + \delta_t \end{aligned}$$

- The functions g and h can be nonlinear functions, modeled using deep neural nets, or GP. Another way is to model these as linear approximations of nonlinear functions (Extended Kalman Filter)

Other Extensions of SSM/LDS

- Nonlinear dynamical systems: Assume state-transition and observation models to be nonlinear

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= g(\boldsymbol{s}_{t-1}) + \epsilon_t \\ \boldsymbol{x}_t | \boldsymbol{s}_t &= h(\boldsymbol{s}_t) + \delta_t \end{aligned}$$

- The functions g and h can be nonlinear functions, modeled using deep neural nets, or GP. Another way is to model these as linear approximations of nonlinear functions (Extended Kalman Filter)
- Switching LDS/SSM: Assumes data to be generated from a mixture of M LDS/SSM

Other Extensions of SSM/LDS

- Nonlinear dynamical systems: Assume state-transition and observation models to be nonlinear

$$\begin{aligned} \boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= g(\boldsymbol{s}_{t-1}) + \epsilon_t \\ \boldsymbol{x}_t | \boldsymbol{s}_t &= h(\boldsymbol{s}_t) + \delta_t \end{aligned}$$

- The functions g and h can be nonlinear functions, modeled using deep neural nets, or GP. Another way is to model these as linear approximations of nonlinear functions (Extended Kalman Filter)
- Switching LDS/SSM: Assumes data to be generated from a mixture of M LDS/SSM
  - For each observation $\boldsymbol{x}_t$, first draw a cluster id $c_t \in \{1, \ldots, M\}$ from a multinoulli

Other Extensions of SSM/LDS

- Nonlinear dynamical systems: Assume state-transition and observation models to be nonlinear

$$
\begin{aligned}
\boldsymbol{s}_t|\boldsymbol{s}_{t-1} &= g(\boldsymbol{s}_{t-1}) + \epsilon_t \\
\boldsymbol{x}_t|\boldsymbol{s}_t &= h(\boldsymbol{s}_t) + \delta_t
\end{aligned}
$$

- The functions g and h can be nonlinear functions, modeled using deep neural nets, or GP. Another way is to model these as linear approximations of nonlinear functions (Extended Kalman Filter)
- Switching LDS/SSM: Assumes data to be generated from a mixture of M LDS/SSM
  - For each observation $\boldsymbol{x}_t$, first draw a cluster id $c_t \in \{1, \ldots, M\}$ from a multinoulli
  - Suppose $c_t = m$. Now generate the observation $\boldsymbol{x}_t$ using the the m-th LDS/SSM

$$
\begin{aligned}
\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, c_t = m &\sim \mathcal{N}(\boldsymbol{s}_t|\mathbf{A}^{(m)}\boldsymbol{s}_{t-1}, \mathbf{Q}^{(m)}) \\
\boldsymbol{x}_t|\boldsymbol{s}_t, c_t = m &\sim \mathcal{N}(\boldsymbol{x}_t|\mathbf{B}^{(m)}\boldsymbol{s}_t, \mathbf{R}^{(m)})
\end{aligned}
$$

Other Extensions of SSM/LDS

- Nonlinear dynamical systems: Assume state-transition and observation models to be nonlinear

$$
\begin{aligned}
\boldsymbol{s}_t | \boldsymbol{s}_{t-1} &= g(\boldsymbol{s}_{t-1}) + \epsilon_t \\
\boldsymbol{x}_t | \boldsymbol{s}_t &= h(\boldsymbol{s}_t) + \delta_t
\end{aligned}
$$

- The functions g and h can be nonlinear functions, modeled using deep neural nets, or GP. Another way is to model these as linear approximations of nonlinear functions (Extended Kalman Filter)
- Switching LDS/SSM: Assumes data to be generated from a mixture of M LDS/SSM
  - For each observation $\boldsymbol{x}_t$, first draw a cluster id $c_t \in \{1, \ldots, M\}$ from a multinoulli
  - Suppose $c_t = m$. Now generate the observation $\boldsymbol{x}_t$ using the the m-th LDS/SSM

$$
\begin{aligned}
\boldsymbol{s}_t | \boldsymbol{s}_{t-1}, c_t = m &\sim \mathcal{N}(\boldsymbol{s}_t | \mathbf{A}^{(m)} \boldsymbol{s}_{t-1}, \mathbf{Q}^{(m)}) \\
\boldsymbol{x}_t | \boldsymbol{s}_t, c_t = m &\sim \mathcal{N}(\boldsymbol{x}_t | \mathbf{B}^{(m)} \boldsymbol{s}_t, \mathbf{R}^{(m)})
\end{aligned}
$$

  - It's a hybrid LDS – the "state" consists of two latent variables $c_t, \boldsymbol{z}_t$ (discrete and continuous)

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably
- These basic models have been extended to more sophisticated models

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably
- These basic models have been extended to more sophisticated models, e.g.,
  - Non-Gaussian LDS, e.g., see Poisson-Gamma dynamical systems by Schein et al (2016)

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably
- These basic models have been extended to more sophisticated models, e.g.,
  - Non-Gaussian LDS, e.g., see Poisson-Gamma dynamical systems by Schein et al (2016)
  - Structured LDS (e.g., the switching LDA that we saw)

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably
- These basic models have been extended to more sophisticated models, e.g.,
  - Non-Gaussian LDS, e.g., see Poisson-Gamma dynamical systems by Schein et al (2016)
  - Structured LDS (e.g., the switching LDA that we saw)
  - Deep LDS (e.g., using a VAE to define $\boldsymbol{z}_n$ to $\boldsymbol{x}_n$ mapping with a recognition model)

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably
- These basic models have been extended to more sophisticated models, e.g.,
  - Non-Gaussian LDS, e.g., see Poisson-Gamma dynamical systems by Schein et al (2016)
  - Structured LDS (e.g., the switching LDA that we saw)
  - Deep LDS (e.g., using a VAE to define $\boldsymbol{z}_n$ to $\boldsymbol{x}_n$ mapping with a recognition model)
  - Combining SSM with recurrent neural nets or LSTM to handle variable length sequences

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably
- These basic models have been extended to more sophisticated models, e.g.,
  - Non-Gaussian LDS, e.g., see Poisson-Gamma dynamical systems by Schein et al (2016)
  - Structured LDS (e.g., the switching LDA that we saw)
  - Deep LDS (e.g., using a VAE to define $\boldsymbol{z}_n$ to $\boldsymbol{x}_n$ mapping with a recognition model)
  - Combining SSM with recurrent neural nets or LSTM to handle variable length sequences
- Inference for HMM is also based on similar principled (e.g., forward and forward-backward algorithm), except that the latent variables are discrete

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably
- These basic models have been extended to more sophisticated models, e.g.,
  - Non-Gaussian LDS, e.g., see Poisson-Gamma dynamical systems by Schein et al (2016)
  - Structured LDS (e.g., the switching LDA that we saw)
  - Deep LDS (e.g., using a VAE to define $\boldsymbol{z}_n$ to $\boldsymbol{x}_n$ mapping with a recognition model)
  - Combining SSM with recurrent neural nets or LSTM to handle variable length sequences
- Inference for HMM is also based on similar principled (e.g., forward and forward-backward algorithm), except that the latent variables are discrete
- The general principle (time-evolving latent variables) can be applied in a wide range of probabilistic models to enable them handle dynamic/time-evolving data

Summary

- SSM/LDS allows modeling non i.i.d. sequential data
- Gaussian assumption on transition/observation models helps inference considerably
- These basic models have been extended to more sophisticated models, e.g.,
  - Non-Gaussian LDS, e.g., see Poisson-Gamma dynamical systems by Schein et al (2016)
  - Structured LDS (e.g., the switching LDA that we saw)
  - Deep LDS (e.g., using a VAE to define $\boldsymbol{z}_n$ to $\boldsymbol{x}_n$ mapping with a recognition model)
  - Combining SSM with recurrent neural nets or LSTM to handle variable length sequences
- Inference for HMM is also based on similar principled (e.g., forward and forward-backward algorithm), except that the latent variables are discrete
- The general principle (time-evolving latent variables) can be applied in a wide range of probabilistic models to enable them handle dynamic/time-evolving data
  - E.g., in LDA, we can make the topic assignments of adjacent words follow a Markov relationship (results in an HMM-LDA type model)

Backup Slides: Kalman Smoothing

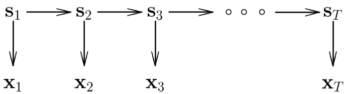
Kalman Smoothing in SSMs

Goal: Infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given <u>all the observations</u> (both past and future)

Kalman Smoothing in SSMs

Goal: Infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given <u>all the observations</u> (both past and future)

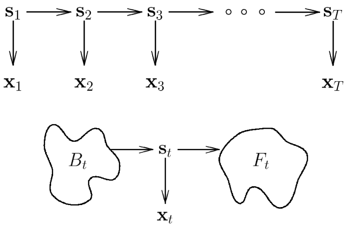
Note that each state variable $\boldsymbol{s}_t$ separates the graph into three independent parts



Kalman Smoothing in SSMs

Goal: Infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given all the observations (both past and future)

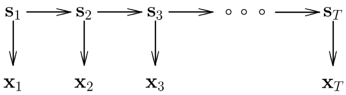
Note that each state variable $\boldsymbol{s}_t$ separates the graph into three independent parts

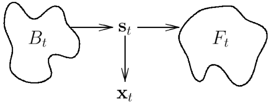


Kalman Smoothing in SSMs

Goal: Infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given <u>all the observations</u> (both past and future)

Note that each state variable $\boldsymbol{s}_t$ separates the graph into three independent parts





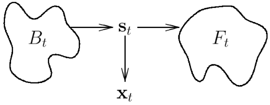
$$
\begin{aligned}
B_t &= \{\mathbf{x}_1..\mathbf{x}_{t-1}, \mathbf{s}_1..\mathbf{s}_{t-1}\} \\
F_t &= \{\mathbf{x}_{t+1}..\mathbf{x}_T, \mathbf{s}_{t+1}..\mathbf{s}_T\}
\end{aligned}
$$

Kalman Smoothing in SSMs

Goal: Infer $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ given all the observations (both past and future)

Note that each state variable $\boldsymbol{s}_t$ separates the graph into three independent parts

$$
\begin{tikzcd}
\mathbf{s}_1 \arrow[r] \arrow[d] & \mathbf{s}_2 \arrow[r] \arrow[d] & \mathbf{s}_3 \arrow[r] \arrow[d] & \circ \circ \circ \arrow[r] & \mathbf{s}_T \arrow[d] \\
\mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & & \mathbf{x}_T
\end{tikzcd}
$$



$$
\begin{aligned}
B_t &= \{\mathbf{x}_1..\mathbf{x}_{t-1}, \mathbf{s}_1..\mathbf{s}_{t-1}\} \\
F_t &= \{\mathbf{x}_{t+1}..\mathbf{x}_T, \mathbf{s}_{t+1}..\mathbf{s}_T\}
\end{aligned}
$$

$$p(B_t, \mathbf{s}_t, \mathbf{x}_t, F_t) = p(B_t, \mathbf{s}_t) p(\mathbf{x}_t|\mathbf{s}_t) p(F_t|\mathbf{s}_t)$$

Kalman Smoothing in SSMs

- Goal: marginal probability $p(\boldsymbol{s}_t|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T)$ of each state (i.e., smoothing)
- Let's look at the joint probability first:

$$\begin{aligned}
p(\mathbf{s}_t,\mathbf{x}_1..\mathbf{x}_T) &= \int_{\mathbf{s}_1..\mathbf{s}_{t-1}}\int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(B_t,\mathbf{s}_t,\mathbf{x}_t,F_t) \\
&= \left(\int_{\mathbf{s}_1..\mathbf{s}_{t-1}} p(B_t,\mathbf{s}_t)\right) p(\mathbf{x}_t|\mathbf{s}_t) \left(\int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(F_t|\mathbf{s}_t)\right) \\
&= p(B_t^x,\mathbf{s}_t)p(\mathbf{x}_t|\mathbf{s}_t)p(F_t^x|\mathbf{s}_t)
\end{aligned}$$

Kalman Smoothing in SSMs

- Goal: marginal probability $p(\boldsymbol{s}_t | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ of each state (i.e., smoothing)
- Let's look at the joint probability first:

$$
\begin{aligned}
p(\mathbf{s}_t, \mathbf{x}_1..\mathbf{x}_T) &= \int_{\mathbf{s}_1..\mathbf{s}_{t-1}} \int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(B_t, \mathbf{s}_t, \mathbf{x}_t, F_t) \\
&= \left(\int_{\mathbf{s}_1..\mathbf{s}_{t-1}} p(B_t, \mathbf{s}_t) \right) p(\mathbf{x}_t|\mathbf{s}_t) \left(\int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(F_t|\mathbf{s}_t) \right) \\
&= p(B_t^x, \mathbf{s}_t) p(\mathbf{x}_t|\mathbf{s}_t) p(F_t^x|\mathbf{s}_t)
\end{aligned}
$$

$$
\begin{aligned}
B_t^x &= \{\mathbf{x}_1..\mathbf{x}_{t-1}\} \\
F_t^x &= \{\mathbf{x}_{t+1}..\mathbf{x}_T\}
\end{aligned}
$$

Kalman Smoothing in SSMs

- Goal: marginal probability $p(\boldsymbol{s}_t|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T)$ of each state (i.e., smoothing)
- Let's look at the joint probability first:

$$
\begin{aligned}
p(\mathbf{s}_t, \mathbf{x}_1..\mathbf{x}_T) &= \int_{\mathbf{s}_1..\mathbf{s}_{t-1}} \int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(B_t, \mathbf{s}_t, \mathbf{x}_t, F_t) \\
&= \left(\int_{\mathbf{s}_1..\mathbf{s}_{t-1}} p(B_t, \mathbf{s}_t)\right) p(\mathbf{x}_t|\mathbf{s}_t) \left(\int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(F_t|\mathbf{s}_t)\right) \\
&= p(B_t^x, \mathbf{s}_t) p(\mathbf{x}_t|\mathbf{s}_t) p(F_t^x|\mathbf{s}_t)
\end{aligned}
$$

$$
B_t^x = \{\mathbf{x}_1..\mathbf{x}_{t-1}\}
$$
$$
F_t^x = \{\mathbf{x}_{t+1}..\mathbf{x}_T\}
$$

$$
\begin{aligned}
\alpha_t(\mathbf{s}_t) &= p(B_t^x, \mathbf{s}_t) p(\mathbf{x}_t|\mathbf{s}_t) = p(B_t^x, \mathbf{x}_t, \mathbf{s}_t) \\
\beta_t(\mathbf{s}_t) &= p(F_t^x|\mathbf{s}_t)
\end{aligned}
$$

Kalman Smoothing in SSMs

- Goal: marginal probability $p(\boldsymbol{s}_t|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ of each state (i.e., smoothing)
- Let's look at the joint probability first:

$$
\begin{aligned}
p(\mathbf{s}_t, \mathbf{x}_1..\mathbf{x}_T) &= \int_{\mathbf{s}_1..\mathbf{s}_{t-1}} \int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(B_t, \mathbf{s}_t, \mathbf{x}_t, F_t) \\
&= \left(\int_{\mathbf{s}_1..\mathbf{s}_{t-1}} p(B_t, \mathbf{s}_t)\right) p(\mathbf{x}_t|\mathbf{s}_t) \left(\int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(F_t|\mathbf{s}_t)\right) \\
&= p(B_t^x, \mathbf{s}_t) p(\mathbf{x}_t|\mathbf{s}_t) p(F_t^x|\mathbf{s}_t)
\end{aligned}
$$

$$
B_t^x = \{\mathbf{x}_1..\mathbf{x}_{t-1}\}
$$
$$
F_t^x = \{\mathbf{x}_{t+1}..\mathbf{x}_T\}
$$

$$
\begin{aligned}
\alpha_t(\mathbf{s}_t) &= p(B_t^x, \mathbf{s}_t) p(\mathbf{x}_t|\mathbf{s}_t) = p(B_t^x, \mathbf{x}_t, \mathbf{s}_t) \\
\beta_t(\mathbf{s}_t) &= p(F_t^x|\mathbf{s}_t)
\end{aligned}
$$

$$
\boxed{p(\mathbf{s}_t, \mathbf{x}_1..\mathbf{x}_T) = \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)}
$$

Kalman Smoothing in SSMs

- Goal: marginal probability $p(\boldsymbol{s}_t|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T)$ of each state (i.e., smoothing)
- Let's look at the joint probability first:

$$
\begin{aligned}
p(\mathbf{s}_t,\mathbf{x}_1..\mathbf{x}_T) &= \int_{\mathbf{s}_1..\mathbf{s}_{t-1}}\int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(B_t,\mathbf{s}_t,\mathbf{x}_t,F_t) \\
&= \left(\int_{\mathbf{s}_1..\mathbf{s}_{t-1}} p(B_t,\mathbf{s}_t)\right) p(\mathbf{x}_t|\mathbf{s}_t)\left(\int_{\mathbf{s}_{t+1}..\mathbf{s}_T} p(F_t|\mathbf{s}_t)\right) \\
&= p(B_t^x,\mathbf{s}_t)p(\mathbf{x}_t|\mathbf{s}_t)p(F_t^x|\mathbf{s}_t)
\end{aligned}
$$

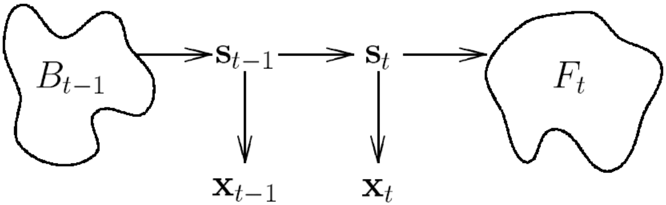
$$
\begin{aligned}
B_t^x &= \{\mathbf{x}_1..\mathbf{x}_{t-1}\} \\
F_t^x &= \{\mathbf{x}_{t+1}..\mathbf{x}_T\}
\end{aligned}
$$

$$
\begin{aligned}
\alpha_t(\mathbf{s}_t) &= p(B_t^x,\mathbf{s}_t)p(\mathbf{x}_t|\mathbf{s}_t) = p(B_t^x,\mathbf{x}_t,\mathbf{s}_t) \\
\beta_t(\mathbf{s}_t) &= p(F_t^x|\mathbf{s}_t)
\end{aligned}
$$

$$
\boxed{p(\mathbf{s}_t,\mathbf{x}_1..\mathbf{x}_T) = \alpha_t(\mathbf{s}_t)\beta_t(\mathbf{s}_t)}
$$

- From the joint, we can compute $p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T) = \sum_{\boldsymbol{s}_t} p(\boldsymbol{s}_t,\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T)$, and $p(\boldsymbol{s}_t|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T)$ using Bayes rule

Estimation via Forward-Backward Recursion



Denote $B_t = B_{t-1} \cup \{\boldsymbol{s}_{t-1}, \boldsymbol{x}_{t-1}\}$ and $F_{t-1} = \{\boldsymbol{s}_t, \boldsymbol{x}_t\} \cup F_t$

Estimation via Forward-Backward Recursion

Denote $B_t = B_{t-1} \cup \{\boldsymbol{s}_{t-1}, \boldsymbol{x}_{t-1}\}$ and $F_{t-1} = \{\boldsymbol{s}_t, \boldsymbol{x}_t\} \cup F_t$

Estimation via Forward-Backward Recursion

Denote $B_t = B_{t-1} \cup \{\boldsymbol{s}_{t-1}, \boldsymbol{x}_{t-1}\}$ and $F_{t-1} = \{\boldsymbol{s}_t, \boldsymbol{x}_t\} \cup F_t$

Can compute α and β recursively

Estimation via Forward-Backward Recursion

Denote $B_t = B_{t-1} \cup \{\boldsymbol{s}_{t-1}, \boldsymbol{x}_{t-1}\}$ and $F_{t-1} = \{\boldsymbol{s}_t, \boldsymbol{x}_t\} \cup F_t$

Can compute α and β recursively

$$
\begin{aligned}
\alpha_t(\mathbf{s}_t) = p(\mathbf{x}_t|\mathbf{s}_t)p(B_t^x, \mathbf{s}_t) &= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(B_{t-1}^x, \mathbf{s}_{t-1} = \mathbf{z}, \mathbf{x}_{t-1}, \mathbf{s}_t) \\
&= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(B_{t-1}^x, \mathbf{s}_{t-1} = \mathbf{z})p(\mathbf{x}_{t-1}|\mathbf{s}_{t-1} = \mathbf{z})p(\mathbf{s}_t|\mathbf{s}_{t-1} = \mathbf{z}) \\
&= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(\mathbf{s}_t|\mathbf{s}_{t-1} = \mathbf{z})\alpha_{t-1}(\mathbf{z})
\end{aligned}
$$

Forward recursion for α

Estimation via Forward-Backward Recursion

Denote $B_t = B_{t-1} \cup \{\boldsymbol{s}_{t-1}, \boldsymbol{x}_{t-1}\}$ and $F_{t-1} = \{\boldsymbol{s}_t, \boldsymbol{x}_t\} \cup F_t$

Can compute α and β recursively

$$
\begin{aligned}
\alpha_t(\mathbf{s}_t) = p(\mathbf{x}_t|\mathbf{s}_t)p(B_t^x, \mathbf{s}_t) &= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(B_{t-1}^x, \mathbf{s}_{t-1} = \mathbf{z}, \mathbf{x}_{t-1}, \mathbf{s}_t) \\
&= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(B_{t-1}^x, \mathbf{s}_{t-1} = \mathbf{z})p(\mathbf{x}_{t-1}|\mathbf{s}_{t-1} = \mathbf{z})p(\mathbf{s}_t|\mathbf{s}_{t-1} = \mathbf{z}) \\
&= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(\mathbf{s}_t|\mathbf{s}_{t-1} = \mathbf{z})\alpha_{t-1}(\mathbf{z})
\end{aligned}
$$

Forward recursion for α

$$
\begin{aligned}
\beta_{t-1}(\mathbf{s}_{t-1}) = p(F_{t-1}^x|\mathbf{s}_{t-1}) &= \int_{\mathbf{z}} p(\mathbf{s}_t = \mathbf{z}, \mathbf{x}_t, F_t^x|\mathbf{s}_{t-1}) \\
&= \int_{\mathbf{z}} p(\mathbf{s}_t = \mathbf{z}|\mathbf{s}_{t-1})p(\mathbf{x}_t|\mathbf{s}_t = \mathbf{z})p(F_t^x|\mathbf{s}_t = \mathbf{z}) \\
&= \int_{\mathbf{z}} p(\mathbf{s}_t = \mathbf{z}|\mathbf{s}_{t-1})p(\mathbf{x}_t|\mathbf{s}_t = \mathbf{z})\beta_t(\mathbf{z})
\end{aligned}
$$

Backward recursion for β

Estimation via Forward-Backward Recursion

Denote $B_t = B_{t-1} \cup \{\boldsymbol{s}_{t-1}, \boldsymbol{x}_{t-1}\}$ and $F_{t-1} = \{\boldsymbol{s}_t, \boldsymbol{x}_t\} \cup F_t$

Can compute α and β recursively

$$\begin{aligned}
\alpha_t(\mathbf{s}_t) = p(\mathbf{x}_t|\mathbf{s}_t)p(B_t^x, \mathbf{s}_t) &= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(B_{t-1}^x, \mathbf{s}_{t-1} = \mathbf{z}, \mathbf{x}_{t-1}, \mathbf{s}_t) \\
&= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(B_{t-1}^x, \mathbf{s}_{t-1} = \mathbf{z})p(\mathbf{x}_{t-1}|\mathbf{s}_{t-1} = \mathbf{z})p(\mathbf{s}_t|\mathbf{s}_{t-1} = \mathbf{z}) \\
&= p(\mathbf{x}_t|\mathbf{s}_t)\int_{\mathbf{z}} p(\mathbf{s}_t|\mathbf{s}_{t-1} = \mathbf{z})\alpha_{t-1}(\mathbf{z})
\end{aligned}$$

Forward recursion for α

$$\begin{aligned}
\beta_{t-1}(\mathbf{s}_{t-1}) = p(F_{t-1}^x|\mathbf{s}_{t-1}) &= \int_{\mathbf{z}} p(\mathbf{s}_t = \mathbf{z}, \mathbf{x}_t, F_t^x|\mathbf{s}_{t-1}) \\
&= \int_{\mathbf{z}} p(\mathbf{s}_t = \mathbf{z}|\mathbf{s}_{t-1})p(\mathbf{x}_t|\mathbf{s}_t = \mathbf{z})p(F_t^x|\mathbf{s}_t = \mathbf{z}) \\
&= \int_{\mathbf{z}} p(\mathbf{s}_t = \mathbf{z}|\mathbf{s}_{t-1})p(\mathbf{x}_t|\mathbf{s}_t = \mathbf{z})\beta_t(\mathbf{z})
\end{aligned}$$

Backward recursion for β

Initialize as $\alpha_1(\boldsymbol{s}_1) = p(\boldsymbol{s}_1)p(\boldsymbol{x}_1|\boldsymbol{s}_1)$ and $\beta_T(\boldsymbol{s}_T) = 1$