

# Machine Learning

C. Reinforcement Learning C.1. State Space Models

#### Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL) Institute for Computer Science University of Hildesheim, Germany

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

1 / 22

・ロト < 団ト < ヨト < ヨト < 国ト < ロト</li>

#### Outline



1. Introduction

2. Inference

3. Learning

もしゃ 予測 ふかん かん きょう

# Syllabus



Tue. 21.10.	(1)	0. Introduction
		A. Supervised Learning
Wed. 22.10.	(2)	A.1 Linear Regression
Tue. 28.10.	(3)	A.2 Linear Classification
Wed. 29.10.	(4)	A.3 Regularization
Tue. 4.11.	(5)	A.4 High-dimensional Data
Wed. 5.11.	(6)	A.5 Nearest-Neighbor Models
Tue. 11.11.	(7)	A.6 Decision Trees
Wed. 12.12.	(8)	A.7 Support Vector Machines
Tue. 18.11.	(9)	A.8 A First Look at Bayesian and Markov Networks
		B. Unsupervised Learning
Wed. 19.11.	(10)	B.1 Clustering
Tue. 25.11.	(11)	B.2 Dimensionality Reduction
Wed. 26.11.	(12)	B.3 Frequent Pattern Mining
		C. Reinforcement Learning
Tue. 2.12.	(13)	C.1 State Space Models
Wed. 3.12.	(14)	C.2 Markov Decision Processes
		< □ > <昼 > < 글 > < 글 = → < 의 = → < < □ > < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - → < - →

Machine Learning 1. Introduction

#### Outline



#### 1. Introduction

2. Inference

3. Learning

シック 単則 エル・エー・エー・

# Hidden Markov Models

- observed variables  $x_1, \ldots, x_M$
- hidden variables  $z_1, \ldots, z_M$

$$p(x_{1:M} \mid z_{1:M}) = p(x_1, \dots, x_M, z_1, \dots, z_M) = p(z_{1:M})p(x_{1:M} \mid z_{1:M})$$
$$= p(z_1) \prod_{m=1}^{M-1} p(z_{m+1} \mid z_m) \prod_{m=1}^{M} p(x_m \mid z_m)$$

- transition model  $p(z_{m+1} | z_m)$
- observation model  $p(x_m \mid z_m)$





Machine Learning 1. Introduction

HMMs



► consist of a discrete-time Markov chain with hidden variables plus an observation model p(x<sub>m</sub> | z<sub>m</sub>)

- 《日》 《聞》 《臣》 《臣》 (三) 『 今へで



- ► consist of a discrete-time Markov chain with hidden variables plus an observation model p(x<sub>m</sub> | z<sub>m</sub>)
- ►  $p(x_{m+1} | x_m)$  can be written as a  $M \times M$  Transition Matrix A

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□



- ► consist of a discrete-time Markov chain with hidden variables plus an observation model p(x<sub>m</sub> | z<sub>m</sub>)
- ►  $p(x_{m+1} | x_m)$  can be written as a  $M \times M$  Transition Matrix A

Observations in an HMM can be **discrete** or **continuous**. Continuous HMM are called **State Space Models** (SSM).



- ► consist of a discrete-time Markov chain with hidden variables plus an observation model p(x<sub>m</sub> | z<sub>m</sub>)
- ►  $p(x_{m+1} | x_m)$  can be written as a  $M \times M$  Transition Matrix A

Observations in an HMM can be **discrete** or **continuous**. Continuous HMM are called **State Space Models** (SSM).

► discrete: observation model is observation matrix  $p(x_m = j | z_m = i) = A(i, j)$ 



- ► consist of a discrete-time Markov chain with hidden variables plus an observation model p(x<sub>m</sub> | z<sub>m</sub>)
- ►  $p(x_{m+1} | x_m)$  can be written as a  $M \times M$  Transition Matrix A

Observations in an HMM can be **discrete** or **continuous**. Continuous HMM are called **State Space Models** (SSM).

- ► discrete: observation model is observation matrix  $p(x_m = j | z_m = i) = A(i, j)$
- ► continuous: conditional Gaussian  $p(x_t \mid z_t = i) = \mathcal{N}(x_t \mid \mu_k, \Sigma_k)$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ★∃▶ ★∃▶ ★目★ 少々で



- ► consist of a discrete-time Markov chain with hidden variables plus an observation model p(x<sub>m</sub> | z<sub>m</sub>)
- ►  $p(x_{m+1} | x_m)$  can be written as a  $M \times M$  Transition Matrix A

Observations in an HMM can be **discrete** or **continuous**. Continuous HMM are called **State Space Models** (SSM).

- ► discrete: observation model is observation matrix  $p(x_m = j | z_m = i) = A(i, j)$
- ► continuous: conditional Gaussian  $p(x_t | z_t = i) = \mathcal{N}(x_t | \mu_k, \Sigma_k)$

HMMs can represent long-range dependencies between observations.

《日》《聞》《臣》《臣》 副言 めんぐ

## Applications



Some Applications for HMMs are:

- automatic speech recognition
- ► activity recognition
- ► gene finding
- • •

#### Outline



1. Introduction

#### 2. Inference

3. Learning

● 20~ 単則 ( ● ● ● ● ● ● ● ● ●

Inference



Inference in HMMs:

・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

### Inference



Inference in HMMs:

► Filtering:

compute  $p(z_m, x_{1:m})$  online or recursively

# Inference

Inference in HMMs:

► Filtering:

compute  $p(z_m, x_{1:m})$  online or recursively

• Prediction:

compute  $p(z_{m+h} \mid x_{1:m}), h > 0$  (horizon)

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

・ロト < 団ト < ヨト < ヨト < 国ト < ロト</li>



## Inference

Universitat

Inference in HMMs:

► Filtering:

compute  $p(z_m, x_{1:m})$  online or recursively

• Prediction:

compute  $p(z_{m+h} \mid x_{1:m}), h > 0$  (horizon) e.g h = 2

$$p(z_{m+2} \mid x_{1:m}) = \sum_{z_{m+1}} \sum_{z_m} p(z_{m+2} \mid z_{m+1}) p(z_{m+1} \mid z_m) p(z_m \mid x_{1:m})$$

- 4日 > 4 個 > - 4 표 > - 4 표 > - 1 표 - 9 9 9

## Inference

Universitat

Inference in HMMs:

► Filtering:

compute  $p(z_m, x_{1:m})$  online or recursively

• Prediction:

compute  $p(z_{m+h} \mid x_{1:m}), h > 0$  (horizon) e.g h = 2

$$p(z_{m+2} \mid x_{1:m}) = \sum_{z_{m+1}} \sum_{z_m} p(z_{m+2} \mid z_{m+1}) p(z_{m+1} \mid z_m) p(z_m \mid x_{1:m})$$

► MAP estimation: compute argmax<sub>z1:M</sub> p(z<sub>1:M</sub> | x<sub>1:M</sub>)

《日》《聞》《臣》《臣》《臣》 (四)





Shaded region is the interval for which we have data

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

[Mur12, fig. 17.11] ▲□→ ▲ ■→ ▲ ■→ ■ ■ → へへ

#### Forward Backward Algorithm



• compute the joint distribution  $p(z_m | x_{1:M})$ 

→□▶→@▶→==>→==>→==>>

#### Forward Backward Algorithm



- compute the joint distribution  $p(z_m | x_{1:M})$
- Use Forward Algorithm to compute  $p(z_m, x_{1:m})$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

・ロト < 団ト < ヨト < ヨト < 国ト < ロト</li>

### Forward Backward Algorithm



- compute the joint distribution  $p(z_m | x_{1:M})$
- Use Forward Algorithm to compute  $p(z_m, x_{1:m})$
- Use Backward Algorithm to compute  $p(x_{m+1:M} | z_m)$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ★∃▶ ★∃▶ ★目★ 少々で

### Forward Backward Algorithm



- compute the joint distribution  $p(z_m | x_{1:M})$
- Use Forward Algorithm to compute  $p(z_m, x_{1:m})$
- Use Backward Algorithm to compute  $p(x_{m+1:M} | z_m)$

► 
$$p(z_m | x_{1:M}) \propto_{z_m} p(z_m, x_{1:M}) = \underbrace{p(x_{m+1:M} | z_m)}_{B} \underbrace{p(z_m, x_{1:m})}_{F}$$
 (normalize and sum over the set)

### Forward Backward Algorithm



- compute the joint distribution  $p(z_m | x_{1:M})$
- Use Forward Algorithm to compute  $p(z_m, x_{1:m})$
- Use Backward Algorithm to compute  $p(x_{m+1:M} | z_m)$
- ►  $p(z_m | x_{1:M}) \propto_{z_m} p(z_m, x_{1:M}) = \underbrace{p(x_{m+1:M} | z_m)}_{B} \underbrace{p(z_m, x_{1:m})}_{F}$  (normalize and sum over the set)
- Assume  $p(x_m \mid z_m), p(z_m \mid z_{m-1}), p(z_1)$  are known

うせん 正則 スポッスポッス セッ

### Forward Algorithm





### Forward Algorithm





$$\alpha_m(z_m) = p(z_m, x_{1:m}) = \sum_{z_{m-1}} p(z_m, z_{m-1}, x_{1:m})$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

・ロト < 団ト < ヨト < ヨト < ロト</li>

#### Forward Algorithm





$$\alpha_m(z_m) = p(z_m, x_{1:m}) = \sum_{z_{m-1}} p(z_m, z_{m-1}, x_{1:m})$$
$$= \sum_{z_{m-1}} p(x_m \mid z_m, z_{m-1}, x_{1:m-1}) p(z_m \mid z_{m-1}, x_{1:m-1}) p(z_{m-1}, x_{1:m-1})$$

◇♪♡ 비로 《로》《토》《唱》《□》

#### Forward Algorithm





$$\begin{aligned} \alpha_m(z_m) &= p(z_m, x_{1:m}) = \sum_{z_{m-1}} p(z_m, z_{m-1}, x_{1:m}) \\ &= \sum_{z_{m-1}} p(x_m \mid z_m, z_{m-1}, x_{1:m-1}) p(z_m \mid z_{m-1}, x_{1:m-1}) p(z_{m-1}, x_{1:m-1}) \\ &= \sum_{z_{m-1}} p(x_m \mid z_m) p(z_m \mid z_{m-1}) \underbrace{p(z_m, x_{1:m-1})}_{\alpha_{m-1}(z_{m-1})} \end{aligned}$$

### Forward Algorithm





$$\begin{aligned} \alpha_m(z_m) &= p(z_m, x_{1:m}) = \sum_{z_{m-1}} p(z_m, z_{m-1}, x_{1:m}) \\ &= \sum_{z_{m-1}} p(x_m \mid z_m, z_{m-1}, x_{1:m-1}) p(z_m \mid z_{m-1}, x_{1:m-1}) p(z_{m-1}, x_{1:m-1}) \\ &= \sum_{z_{m-1}} p(x_m \mid z_m) p(z_m \mid z_{m-1}) \underbrace{p(z_m, x_{1:m-1})}_{\alpha_{m-1}(z_{m-1})} \\ \alpha_1(z_1) &= p(z_1, x_1) = p(z_1) p(x_1 \mid z_1) \end{aligned}$$

# Backward Algorithm



Given  $x_1, \ldots, x_M$ : Compute  $p(x_{m+1:M} | z_m)$  for all m and  $z_m$ .

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany



### Backward Algorithm



Given  $x_1, ..., x_M$ : Compute  $p(x_{m+1:M} | z_m)$  for all m and  $z_m$ .  $\beta_m(z_m) = p(x_{m+1:M} | z_m) = \sum_{z_{m+1}} p(x_{m+1:M}, z_{m+1} | z_m)$ 

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

▲母 → ▲目 → モ → 目目 ろくで



#### Backward Algorithm



Given 
$$x_1, ..., x_M$$
:  
Compute  $p(x_{m+1:M} | z_m)$  for all  $m$  and  $z_m$ .  
 $\beta_m(z_m) = p(x_{m+1:M} | z_m) = \sum_{z_{m+1}} p(x_{m+1:M}, z_{m+1} | z_m)$   
 $= \sum_{z_{m+1}} p(x_{m+2:M} | z_{m+1}, z_m, x_{m+1}) p(x_{m+1} | z_{m+1}, z_m) p(z_{m+1} | z_m)$ 

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

・ロト < 団ト < ヨト < ヨト < ロト</li>



#### Backward Algorithm



Given 
$$x_1, ..., x_M$$
:  
Compute  $p(x_{m+1:M} | z_m)$  for all  $m$  and  $z_m$ .  
 $\beta_m(z_m) = p(x_{m+1:M} | z_m) = \sum_{z_{m+1}} p(x_{m+1:M}, z_{m+1} | z_m)$   
 $= \sum_{z_{m+1}} p(x_{m+2:M} | z_{m+1}, z_m, x_{m+1}) p(x_{m+1} | z_{m+1}, z_m) p(z_{m+1} | z_m)$   
 $= \sum_{z_{m+1}} \underbrace{p(x_{m+2:M} | z_{m+1})}_{\beta_{m+1}(z_{m+1})} p(x_{m+1} | z_{m+1}) p(z_{m+1} | z_m)$ 

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

・ロト < 団ト < ヨト < ヨト < ロト</li>



#### Backward Algorithm



Given 
$$x_1, \dots, x_M$$
:  
Compute  $p(x_{m+1:M} \mid z_m)$  for all  $m$  and  $z_m$ .  
 $\beta_m(z_m) = p(x_{m+1:M} \mid z_m) = \sum_{z_{m+1}} p(x_{m+1:M}, z_{m+1} \mid z_m)$   
 $= \sum_{z_{m+1}} p(x_{m+2:M} \mid z_{m+1}, z_m, x_{m+1}) p(x_{m+1} \mid z_{m+1}, z_m) p(z_{m+1} \mid z_m)$   
 $= \sum_{z_{m+1}} \underbrace{p(x_{m+2:M} \mid z_{m+1})}_{\beta_{m+1}(z_{m+1})} p(x_{m+1} \mid z_{m+1}) p(z_{m+1} \mid z_m)$   
 $\beta_M(z_M) = 1, \quad \forall z_M$ 



# Viterbi Algorithm





Given:  $x_1, \ldots, x_M$ Assume distributions are known.

もしゃ 正明 ふばやえばや ふむやくしゃ

# Viterbi Algorithm





Given:  $x_1, \ldots, x_M$ Assume distributions are known. Compute  $z^* = \arg \max_{z_{1:M}} p(z_{1:M} | x_{1:M})$ 

もしゃ 正明 ふばやえばや ふむやくしゃ
#### Viterbi Algorithm





Given: 
$$x_1, \ldots, x_M$$
  
Assume distributions are known.  
Compute  $z^* = \arg \max_{z_{1:M}} p(z_{1:M} | x_{1:M})$   
Note:  
 $\arg \max_z p(z | x) = \arg \max_z p(z, x)$ 

《日》《聞》《臣》《臣》 副言 めんぐ

#### Universiter Hildeshein

#### Viterbi Algorithm

$$\delta_m(z_m) = \max_{z_{1:m-1}} p(z_{1:m}, x_{1:k})$$

シック 単語 《王》 《王》 《四》 《四》

Viterbi Algorithm



$$\delta_m(z_m) = \max_{z_{1:m-1}} p(z_{1:m}, x_{1:k})$$
  
=  $\max_{z_{1:m-1}} p(x_m \mid z_m) p(z_m \mid z_{m-1}) p(z_{1:m-1}, x_{1:m-1})$ 

◇□▶ <@▶ < E▶ < E▶ < E⊨ <0.0</p>

δ

#### Viterbi Algorithm



$$m(z_m) = \max_{z_{1:m-1}} p(z_{1:m}, x_{1:k})$$
  
=  $\max_{z_{1:m-1}} p(x_m \mid z_m) p(z_m \mid z_{m-1}) p(z_{1:m-1}, x_{1:m-1})$   
=  $\max_{z_{m-1}} \left( p(x_m \mid z_m) p(z_m \mid z_{m-1}) \underbrace{\max_{z_{1:m-2}} p(z_{1:m-1}, x_{1:m-1})}_{\delta_{m-1}(z_{m-1})} \right)$ 

# Shiversized

#### Viterbi Algorithm

$$\delta_{m}(z_{m}) = \max_{z_{1:m-1}} p(z_{1:m}, x_{1:k})$$
  
=  $\max_{z_{1:m-1}} p(x_{m} | z_{m}) p(z_{m} | z_{m-1}) p(z_{1:m-1}, x_{1:m-1})$   
=  $\max_{z_{m-1}} \left( p(x_{m} | z_{m}) p(z_{m} | z_{m-1}) \underbrace{\max_{z_{1:m-2}} p(z_{1:m-1}, x_{1:m-1})}_{\delta_{m-1}(z_{m-1})} \right)$ 

We also keep track of the maximizing sequence in each step

# Shiversiter

#### Viterbi Algorithm

$$\delta_{m}(z_{m}) = \max_{z_{1:m-1}} p(z_{1:m}, x_{1:k})$$
  
=  $\max_{z_{1:m-1}} p(x_{m} | z_{m}) p(z_{m} | z_{m-1}) p(z_{1:m-1}, x_{1:m-1})$   
=  $\max_{z_{m-1}} \left( p(x_{m} | z_{m}) p(z_{m} | z_{m-1}) \underbrace{\max_{z_{1:m-2}} p(z_{1:m-1}, x_{1:m-1})}_{\delta_{m-1}(z_{m-1})} \right)$ 

We also keep track of the maximizing sequence in each step  $a_m(z_m) = \arg \max_i \delta_{m-1}(i)p(z_m = j \mid z_{m-1} = i)p(x_m \mid z_m = j)$ 

《日》《四》《臣》《臣》 三世 ろくぐ

# Shiversiter

#### Viterbi Algorithm

$$\delta_{m}(z_{m}) = \max_{z_{1:m-1}} p(z_{1:m}, x_{1:k})$$
  
=  $\max_{z_{1:m-1}} p(x_{m} | z_{m}) p(z_{m} | z_{m-1}) p(z_{1:m-1}, x_{1:m-1})$   
=  $\max_{z_{m-1}} \left( p(x_{m} | z_{m}) p(z_{m} | z_{m-1}) \underbrace{\max_{z_{1:m-2}} p(z_{1:m-1}, x_{1:m-1})}_{\delta_{m-1}(z_{m-1})} \right)$ 

We also keep track of the maximizing sequence in each step  $a_m(z_m) = \arg \max_i \delta_{m-1}(i)p(z_m = j | z_{m-1} = i)p(x_m | z_m = j)$ most probable final state  $z_M^*$  $z_M^* = \arg \max_i \delta_M(i)$ 

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□

# Shiversiter

#### Viterbi Algorithm

$$\delta_{m}(z_{m}) = \max_{z_{1:m-1}} p(z_{1:m}, x_{1:k})$$

$$= \max_{z_{1:m-1}} p(x_{m} \mid z_{m}) p(z_{m} \mid z_{m-1}) p(z_{1:m-1}, x_{1:m-1})$$

$$= \max_{z_{m-1}} \left( p(x_{m} \mid z_{m}) p(z_{m} \mid z_{m-1}) \underbrace{\max_{z_{1:m-2}} p(z_{1:m-1}, x_{1:m-1})}_{\delta_{m-1}(z_{m-1})} \right)$$

We also keep track of the maximizing sequence in each step  $a_m(z_m) = \arg \max_i \delta_{m-1}(i)p(z_m = j \mid z_{m-1} = i)p(x_m \mid z_m = j)$ most probable final state  $z_M^*$   $z_M^* = \arg \max_i \delta_M(i)$ traceback:  $z_m^* = a_{m+1}(z_{m+1}^*)$ 

#### Example





[Mur12, fig. 17.3] 《 다 ▷ 《 문 ▷ 《 코 ▷ 《 코 ▷ 코 = ^ 오 < ~

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

13 / 22



 State Space Models (SSM) are like HMM, except hidden states are continuous



- State Space Models (SSM) are like HMM, except hidden states are continuous
- ► special cas LG-SSM, all the CPDs are linear-Gaussian

- (日本) - (11)



- State Space Models (SSM) are like HMM, except hidden states are continuous
- ► special cas LG-SSM, all the CPDs are linear-Gaussian
- ► Transition model and observation model are linear function

・ロト・西ト・王ト・王ト 沙々の



- State Space Models (SSM) are like HMM, except hidden states are continuous
- ► special cas LG-SSM, all the CPDs are linear-Gaussian
- ► Transition model and observation model are linear function

$$\begin{split} z_m &= A_m z_{m-1} + \epsilon_m, \quad \epsilon_m \text{ system noise (Gaussian)} ,\\ \epsilon_m &\sim \mathcal{N}(0, Q_m) \\ y_m &= C_m z_m + \delta_m, \quad \delta_m \text{ observation noise (Gaussian)} ,\\ \epsilon_m &\sim \mathcal{N}(0, R_m) \end{split}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□

#### Inference Kalman Filter



► initial belief state Gaussian  $p(z_1) = \mathcal{N}(\mu_{1|0}, \Sigma_{1|0})$  then  $p(z_m | y_{1:m}) = \mathcal{N}(\mu_m, \Sigma_m)$  are Gaussian



- ► initial belief state Gaussian  $p(z_1) = \mathcal{N}(\mu_{1|0}, \Sigma_{1|0})$  then  $p(z_m | y_{1:m}) = \mathcal{N}(\mu_m, \Sigma_m)$  are Gaussian
- online case is analogous to Forward Algorithm for HMM

- 《日》 《聞》 《臣》 《臣》 三世 '今へぐ



- ► initial belief state Gaussian  $p(z_1) = \mathcal{N}(\mu_{1|0}, \Sigma_{1|0})$  then  $p(z_m | y_{1:m}) = \mathcal{N}(\mu_m, \Sigma_m)$  are Gaussian
- ► online case is analogous to Forward Algorithm for HMM
- ► offline case is analogous to Forward-Backward-Algorithm for HMM



- ► initial belief state Gaussian  $p(z_1) = \mathcal{N}(\mu_{1|0}, \Sigma_{1|0})$  then  $p(z_m | y_{1:m}) = \mathcal{N}(\mu_m, \Sigma_m)$  are Gaussian
- ► online case is analogous to Forward Algorithm for HMM
- offline case is analogous to Forward-Backward-Algorithm for HMM
- ► Kalman Filter: algorithm for exact Bayesian filtering for LG-SSM



- ► initial belief state Gaussian  $p(z_1) = \mathcal{N}(\mu_{1|0}, \Sigma_{1|0})$  then  $p(z_m | y_{1:m}) = \mathcal{N}(\mu_m, \Sigma_m)$  are Gaussian
- ► online case is analogous to Forward Algorithm for HMM
- ► offline case is analogous to Forward-Backward-Algorithm for HMM
- ► Kalman Filter: algorithm for exact Bayesian filtering for LG-SSM
- ▶ marginal posterior at time *m*

$$p(z_m \mid y_{1:m}) = \mathcal{N}(z_m \mid \mu_m, \Sigma_m)$$

シック 비로 《파》《파》《西》 《日》

Algorithm Prediction Step:



Algorithm Prediction Step:



$$p(z_m \mid y_{1:m-1}) = \mathcal{N}(z_m \mid \mu_{m|m-1}, \Sigma_{m|m-1})$$

▲□▶ ▲圖▶ ▲ 불▶ ▲ 불▶ ▲ 특별 ● 의 Q @

Algorithm Prediction Step:



$$p(z_m \mid y_{1:m-1}) = \mathcal{N}(z_m \mid \mu_{m \mid m-1}, \Sigma_{m \mid m-1})$$
$$\mu_{m \mid m-1} = A_m \mu_{m-1}$$
$$\Sigma_{m \mid m-1} = A_m \Sigma_{m-1} A_m^T + Q_m$$

シック 単正 < 王 > < 王 > < 国 > < 国 > < 国 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 四 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

Algorithm Prediction Step:



$$p(z_m \mid y_{1:m-1}) = \mathcal{N}(z_m \mid \mu_{m|m-1}, \Sigma_{m|m-1})$$
$$\mu_{m|m-1} = A_m \mu_{m-1}$$
$$\Sigma_{m|m-1} = A_m \Sigma_{m-1} A_m^T + Q_m$$

Update Step:

うせん 正則 スポッスポッスポッスロッ

Algorithm Prediction Step:



$$p(z_m \mid y_{1:m-1}) = \mathcal{N}(z_m \mid \mu_{m|m-1}, \Sigma_{m|m-1})$$
$$\mu_{m|m-1} = A_m \mu_{m-1}$$
$$\Sigma_{m|m-1} = A_m \Sigma_{m-1} A_m^T + Q_m$$

Update Step:

$$p(z_m | y_m, y_{1:m-1}) \propto p(y_m | z_m)p(z_m | y_{1:m-1})$$

・日本・西本・山田・山田・山市・

Algorithm Prediction Step:



$$p(z_m \mid y_{1:m-1}) = \mathcal{N}(z_m \mid \mu_{m|m-1}, \Sigma_{m|m-1})$$
$$\mu_{m|m-1} = A_m \mu_{m-1}$$
$$\Sigma_{m|m-1} = A_m \Sigma_{m-1} A_m^T + Q_m$$

Update Step:

$$p(z_m \mid y_m, y_{1:m-1}) \propto p(y_m \mid z_m) p(z_m \mid y_{1:m-1}) \\ p(z_m \mid y_{1:m}) = \mathcal{N}(z_m \mid \mu_m, \Sigma_m)$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

・ロト < 団ト < ヨト < ヨト < ロト</li>

Algorithm Prediction Step:



$$p(z_m \mid y_{1:m-1}) = \mathcal{N}(z_m \mid \mu_{m|m-1}, \Sigma_{m|m-1})$$
$$\mu_{m|m-1} = A_m \mu_{m-1}$$
$$\Sigma_{m|m-1} = A_m \Sigma_{m-1} A_m^T + Q_m$$

Update Step:

$$p(z_m \mid y_m, y_{1:m-1}) \propto p(y_m \mid z_m)p(z_m \mid y_{1:m-1})$$

$$p(z_m \mid y_{1:m}) = \mathcal{N}(z_m \mid \mu_m, \Sigma_m)$$

$$\mu_{m|m-1} = \mu_{m|m-1} + \mathcal{K}_m r_m$$

$$\Sigma_{m|m-1} = (I - \mathcal{K}_m C_m) \Sigma_{m|m-1}$$

もうてい 正則 ふかくふやふ きやくしゃ





with  $r_m$  residual or innovation

Algorithm



with  $r_m$  residual or innovation

$$r_m \triangleq y_m - \hat{y}_m$$
  
 $\hat{y}_m \triangleq C_m \mu_{m|m-1}$ 

うとの 単同 《日》《日》 《日》

Algorithm



with  $r_m$  residual or innovation

$$r_m \triangleq y_m - \hat{y}_m$$
  
 $\hat{y}_m \triangleq C_m \mu_{m|m-1}$ 

and K<sub>m</sub> Kalman gain matrix

$$K_m \triangleq \Sigma_{m|m-1} C_m^T S_m^{-1}$$
$$S_m \triangleq C_m \Sigma_{m|m-1} C_m^T + R_m$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

・ロト < 団ト < ヨト < ヨト < ロト</li>

Algorithm



with  $r_m$  residual or innovation

$$r_m \triangleq y_m - \hat{y}_m$$
  
 $\hat{y}_m \triangleq C_m \mu_{m|m-1}$ 

and K<sub>m</sub> Kalman gain matrix

$$K_m \triangleq \Sigma_{m|m-1} C_m^T S_m^{-1}$$
$$S_m \triangleq C_m \Sigma_{m|m-1} C_m^T + R_m$$

All quantities for algorithm

### Kalman Smoothing Algorithm



Analogous to Forward-Backward Algorithm for HMM Backwards equations:

(4 미 ) (4 명 ) (4 명 ) (4 명 ) (4 미 )

### Kalman Smoothing Algorithm

Analogous to Forward-Backward Algorithm for HMM Backwards equations:

$$p(z_m \mid y_{1:M}) = \mathcal{N}(\mu_{m\mid M}, \Sigma_{m\mid M})$$
$$\mu_{m\mid M} = \mu_m + J_m(\mu_{m+1\mid M} - \mu_{m+1\mid m})$$
$$\Sigma_{m\mid M} = \Sigma_m + J_m(\Sigma_{m+1\mid M} - \Sigma_{m+1\mid m})J_m^T$$
$$J_m \triangleq \Sigma_m A_{m+1}^T \Sigma_{m+1\mid m}^{-1}$$



### Kalman Smoothing Algorithm

Analogous to Forward-Backward Algorithm for HMM Backwards equations:

$$p(z_m \mid y_{1:M}) = \mathcal{N}(\mu_{m|M}, \Sigma_{m|M})$$
$$\mu_{m|M} = \mu_m + J_m(\mu_{m+1|M} - \mu_{m+1|m})$$
$$\Sigma_{m|M} = \Sigma_m + J_m(\Sigma_{m+1|M} - \Sigma_{m+1|m})J_m^T$$
$$J_m \triangleq \Sigma_m A_{m+1}^T \Sigma_{m+1|m}^{-1}$$

J<sub>m</sub> is the **backwards Kalman gain matrix** 

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□



### Kalman Smoothing Algorithm

Analogous to Forward-Backward Algorithm for HMM Backwards equations:

$$p(z_m \mid y_{1:M}) = \mathcal{N}(\mu_{m\mid M}, \Sigma_{m\mid M})$$
$$\mu_{m\mid M} = \mu_m + J_m(\mu_{m+1\mid M} - \mu_{m+1\mid m})$$
$$\Sigma_{m\mid M} = \Sigma_m + J_m(\Sigma_{m+1\mid M} - \Sigma_{m+1\mid m})J_m^T$$
$$J_m \triangleq \Sigma_m A_{m+1}^T \Sigma_{m+1\mid m}^{-1}$$

 $J_m$  is the **backwards Kalman gain matrix** Initialized with  $\mu_m$  and  $\Sigma_m$  from the Kalman Filter

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□



#### Outline



1. Introduction

2. Inference

#### 3. Learning

#### Learning HMMs



How to estimate the parameters  $\theta = (\pi, A, B)$ .

#### Learning HMMs



How to estimate the parameters  $\theta = (\pi, A, B)$ .

 $\pi(i) = p(z_1 = i)$  initial state distribution,  $A(i,j) = p(z_m = j \mid z_{m-1} = i)$  transition matrix and *B* are the parameters of the class-conditional densities  $p(x_m \mid z_m = j)$ .
#### Learning HMMs



How to estimate the parameters  $\theta = (\pi, A, B)$ .

 $\pi(i) = p(z_1 = i)$  initial state distribution,  $A(i,j) = p(z_m = j \mid z_{m-1} = i)$  transition matrix and B are the parameters of the class-conditional densities  $p(x_m \mid z_m = j)$ . Algorithm:

### Learning HMMs



How to estimate the parameters  $\theta = (\pi, A, B)$ .

 $\pi(i) = p(z_1 = i)$  initial state distribution,  $A(i,j) = p(z_m = j \mid z_{m-1} = i)$  transition matrix and B are the parameters of the class-conditional densities  $p(x_m \mid z_m = j)$ .

Algorithm: Baum-Welch-Algorithm (EM-Learning)

《日》《四》《臣》《臣》 三世 ろくぐ

Machine Learning 3. Learning

#### Baum-Welch-Algorithm



 Uses EM-Algorithm to find the maximum likelihood estimate of the parameters of a HMM



- Uses EM-Algorithm to find the maximum likelihood estimate of the parameters of a HMM
- given observed feature vectors

- (日本) (四本) (日本) (日本) (日本) (日本)



- Uses EM-Algorithm to find the maximum likelihood estimate of the parameters of a HMM
- given observed feature vectors
- finds local maximum for  $\theta^* = \max_{\theta} p(X \mid \theta)$

- 《日》 《圖》 《王》 《王》 관달 '오오오



- Uses EM-Algorithm to find the maximum likelihood estimate of the parameters of a HMM
- ► given observed feature vectors
- finds local maximum for  $\theta^* = \max_{\theta} p(X \mid \theta)$
- ► Set θ = (A, B, π) with random initial conditions (if not no prior information is known)

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ★∃▶ ★∃▶ ★目★ 少々で



- Uses EM-Algorithm to find the maximum likelihood estimate of the parameters of a HMM
- ► given observed feature vectors
- finds local maximum for  $\theta^* = \max_{\theta} p(X \mid \theta)$
- ► Set θ = (A, B, π) with random initial conditions (if not no prior information is known)
- Use Forward an d Backwards Algorithms to calculate temporary variables:



- Uses EM-Algorithm to find the maximum likelihood estimate of the parameters of a HMM
- given observed feature vectors
- finds local maximum for  $\theta^* = \max_{\theta} p(X \mid \theta)$
- ► Set θ = (A, B, π) with random initial conditions (if not no prior information is known)

a

## Baum-Welch-Algorithm



- Uses EM-Algorithm to find the maximum likelihood estimate of the parameters of a HMM
- given observed feature vectors
- finds local maximum for  $\theta^* = \max_{\theta} p(X \mid \theta)$
- ► Set θ = (A, B, π) with random initial conditions (if not no prior information is known)

$$\begin{aligned} \xi_{ij}(m) &= p(z_m = i, z_{m+1} = j \mid x_{1:m}, \theta) \\ &= \frac{\alpha_i(m)a_{ij}\beta_j(m+1)b_j(x_{m+1})}{\sum_m \alpha_m(m)\beta_m(m)} \\ \\ ij &= p(z_m = j \mid z_{m-1} = i), p(x_{m+1} \mid z_m = j) = b_j(x_{m+1}) \end{aligned}$$



- Uses EM-Algorithm to find the maximum likelihood estimate of the parameters of a HMM
- ► given observed feature vectors
- finds local maximum for  $\theta^* = \max_{\theta} p(X \mid \theta)$
- ► Set θ = (A, B, π) with random initial conditions (if not no prior information is known)

$$\xi_{ij}(m) = p(z_m = i, z_{m+1} = j \mid x_{1:m}, \theta)$$
$$= \frac{\alpha_i(m)a_{ij}\beta_j(m+1)b_j(x_{m+1})}{\sum_m \alpha_m(m)\beta_m(m)}$$

 $a_{ij} = p(z_m = j \mid z_{m-1} = i), p(x_{m+1} \mid z_m = j) = b_j(x_{m+1})$  $\bullet \ \theta \text{ can now be updated using M-Step}$ 

Machine Learning 3. Learning

#### EM for LG-SSM



▶ if we only observe output sequence we can use EM

### EM for LG-SSM



- ▶ if we only observe output sequence we can use EM
- ► Quite similar to Baum-Welch Algorithm for HMMs

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

◆□▶ ◆□▶ ◆□▶ ◆□▶ ◆□ ● ◆○

### EM for LG-SSM



- ► if we only observe output sequence we can use EM
- ► Quite similar to Baum-Welch Algorithm for HMMs
- Except use Kalman Smoothing instead of forward-backwards in the E step

### EM for LG-SSM



- ► if we only observe output sequence we can use EM
- ► Quite similar to Baum-Welch Algorithm for HMMs
- Except use Kalman Smoothing instead of forward-backwards in the E step
- ▶ and use different calculation in the M step

うせん 判所 《田》《田》《田》《日》



#### Further Readings

▶ [Mur12, chapter 17,18].

・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・
・

#### References



Kevin P. Murphy.

*Machine learning: a probabilistic perspective.* The MIT Press, 2012.

<ロ> < 団> < 団> < 三> < 三> < 三</p>