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

Outline

1. Introduction

2. Inference

3. Learning

Syllabus

Outline

1. Introduction

2. Inference

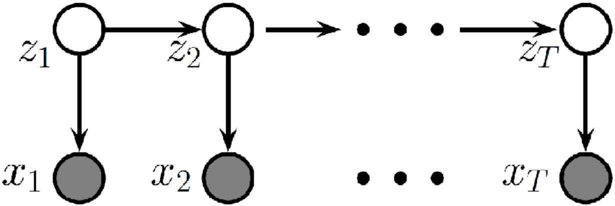
3. Learning

Hidden Markov Models

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

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

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



[Mur12, fig. 10.4]

HMMs

- consist of a discrete-time Markov chain with hidden variables plus an **observation model** $p(x_m \mid z_m)$

HMMs

- consist of a discrete-time Markov chain with hidden variables plus an **observation model** $p(x_m \mid z_m)$
- $p(x_{m+1} \mid x_m)$ can be written as a $M \times M$ **Transition Matrix** A

HMMs

- consist of a discrete-time Markov chain with hidden variables plus an **observation model** $p(x_m \mid z_m)$
- $p(x_{m+1} \mid x_m)$ 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).

HMMs

- consist of a discrete-time Markov chain with hidden variables plus an **observation model** $p(x_m \mid z_m)$
- $p(x_{m+1} \mid x_m)$ 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
  $p(x_m = j \mid z_m = i) = A(i,j)$

HMMs

- consist of a discrete-time Markov chain with hidden variables plus an **observation model** $p(x_m \mid z_m)$
- $p(x_{m+1} \mid x_m)$ 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
  $p(x_m = j \mid z_m = i) = A(i, j)$
- continuous: conditional Gaussian
  $p(x_t \mid z_t = i) = \mathcal{N}(x_t \mid \mu_k, \Sigma_k)$

HMMs

- consist of a discrete-time Markov chain with hidden variables plus an **observation model** $p(x_m \mid z_m)$
- $p(x_{m+1} \mid x_m)$ 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 $p(x_m = j \mid z_m = i) = A(i, j)$
- continuous: conditional Gaussian $p(x_t \mid z_t = i) = \mathcal{N}(x_t \mid \mu_k, \Sigma_k)$

HMMs can represent long-range dependencies between observations.

Applications

Some Applications for HMMs are:

- automatic speech recognition
- activity recognition
- gene finding
- ...

Outline

1. Introduction

2. Inference

3. Learning

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)

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)
  e.g $h = 2$

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

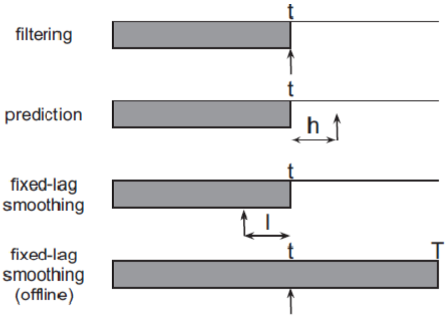
Inference

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)
  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 $\text{argmax}_{z_{1:M}} p(z_{1:M} \mid x_{1:M})$



Shaded region is the interval for which we have data

[Mur12, fig. 17.11]

Forward Backward Algorithm

- compute the joint distribution $p(z_m \mid x_{1:M})$

Forward Backward Algorithm

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

Forward Backward Algorithm

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

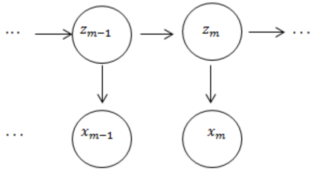
Forward Backward Algorithm

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

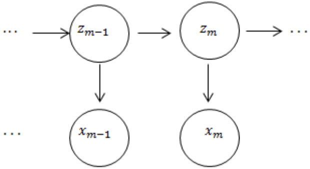
Forward Backward Algorithm

- compute the joint distribution $p(z_m \mid x_{1:M})$
- Use Forward Algorithm to compute $p(z_m, x_{1:m})$
- Use Backward Algorithm to compute $p(x_{m+1:M} \mid z_m)$
- $p(z_m \mid x_{1:M}) \propto_{z_m} p(z_m, x_{1:M}) = \underbrace{p(x_{m+1:M} \mid z_m)}_{\text{B}} \underbrace{p(z_m, x_{1:m})}_{\text{F}}$ (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

Forward Algorithm

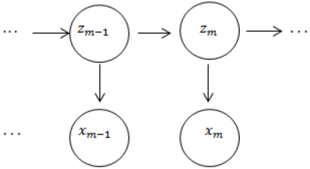


Forward Algorithm



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

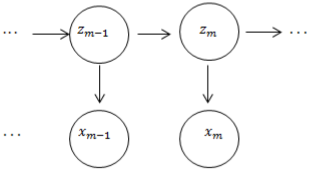
Forward Algorithm



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

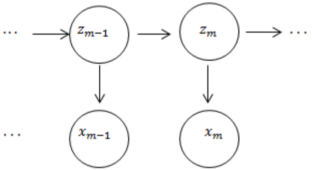
$$= \sum_{z_{m-1}} p(x_m \mid z_m, z_{m-1}, x_{1:m-1}) p(z_m \mid z_{m-1}, x_{1:m-1}) p(z_{m-1}, x_{1:m-1})$$

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



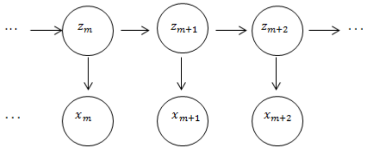
$$\alpha_m(z_m) = p(z_m, x_{1:m}) = \sum_{z_{m-1}} p(z_m, z_{m-1}, x_{1:m})$$

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

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

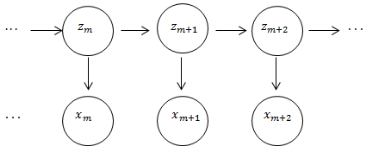
$$\alpha_1(z_1) = p(z_1, x_1) = p(z_1) p(x_1 \mid z_1)$$

Backward Algorithm



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

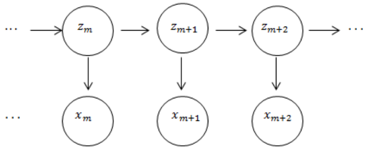
Backward Algorithm



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

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

Backward Algorithm

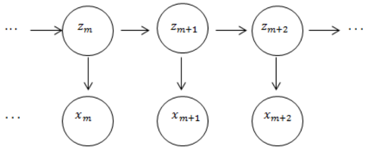


Given $x_1, \ldots, x_M$:
Compute $p(x_{m+1:M} \mid z_m)$ for all m and z_m.

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

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

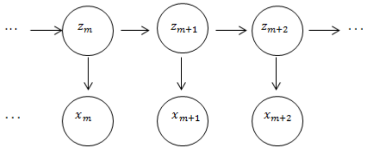
Backward Algorithm



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

$$\begin{aligned}
\beta_m(z_m) = p(x_{m+1:M} \mid z_m) &= \sum_{z_{m+1}} p(x_{m+1:M}, z_{m+1} \mid z_m) \\
&= \sum_{z_{m+1}} p(x_{m+2:M} \mid z_{m+1}, z_m, x_{m+1}) p(x_{m+1} \mid z_{m+1}, z_m) p(z_{m+1} \mid z_m) \\
&= \sum_{z_{m+1}} \underbrace{p(x_{m+2:M} \mid z_{m+1})}_{\beta_{m+1}(z_{m+1})} p(x_{m+1} \mid z_{m+1}) p(z_{m+1} \mid z_m)
\end{aligned}$$

Backward Algorithm



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

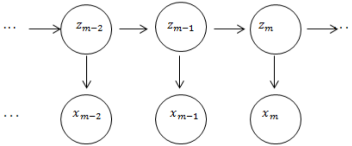
$$\beta_m(z_m) = p(x_{m+1:M} \mid z_m) = \sum_{z_{m+1}} p(x_{m+1:M}, z_{m+1} \mid z_m)$$

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

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

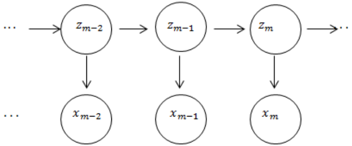
$$\beta_M(z_M) = 1, \quad \forall z_M$$

Viterbi Algorithm



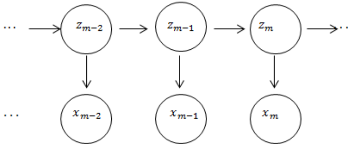
Given: $x_1, \ldots, x_M$
Assume distributions are known.

Viterbi Algorithm



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

Viterbi Algorithm



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

Viterbi Algorithm

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

Viterbi Algorithm

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

Viterbi Algorithm

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

Viterbi Algorithm

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

We also keep track of the maximizing sequence in each step

Viterbi Algorithm

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

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

Viterbi Algorithm

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

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^*
$z_M^* = \arg\max_i \delta_M(i)$

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

$a_m(z_m) = \arg\max_i \delta_{m-1}(i) p(z_m = j \mid z_{m-1} = i) p(x_m \mid z_m = j)$

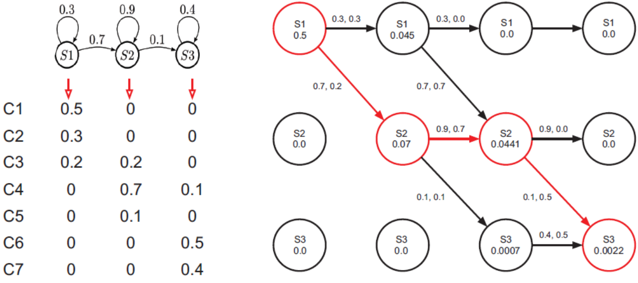
most probable final state z^*_M

$z^*_M = \arg\max_i \delta_M(i)$

traceback:

$z^*_m = a_{m+1}(z^*_{m+1})$

Example



(a) (b)

[Mur12, fig. 17.3]

Kalman Filter

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

Kalman Filter

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

Kalman Filter

- 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

Kalman Filter

- 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
-

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

Inference Kalman Filter

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

Inference Kalman Filter

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

Inference Kalman Filter

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

Inference Kalman Filter

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

Inference Kalman Filter

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

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

Algorithm

Prediction Step:

Algorithm

Prediction Step:

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

Algorithm

Prediction Step:

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

$$\mu_{m|m-1} = A_m \mu_{m-1}$$

$$\Sigma_{m|m-1} = A_m \Sigma_{m-1} A_m^T + Q_m$$

Algorithm

Prediction Step:

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

Update Step:

Algorithm

Prediction Step:

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

Update Step:

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

Algorithm

Prediction Step:

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

Update Step:

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

Algorithm

Prediction Step:

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

Update Step:

$$
\begin{aligned}
p(z_m \mid y_m, y_{1:m-1}) &\propto p(y_m \mid z_m) p(z_m \mid y_{1:m-1}) \\
p(z_m \mid y_{1:m}) &= \mathcal{N}(z_m \mid \mu_m, \Sigma_m) \\
\mu_{m|m-1} &= \mu_{m|m-1} + K_m r_m \\
\Sigma_{m|m-1} &= (I - K_m C_m) \Sigma_{m|m-1}
\end{aligned}
$$

Algorithm

with r_m **residual** or **innovation**

Algorithm

with r_m **residual** or **innovation**

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

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_m **Kalman gain matrix**

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

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_m **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:

Kalman Smoothing Algorithm

Analogous to Forward-Backward Algorithm for HMM
Backwards equations:

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

Kalman Smoothing Algorithm

Analogous to Forward-Backward Algorithm for HMM
Backwards equations:

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

J_m is the **backwards Kalman gain matrix**

Kalman Smoothing Algorithm

Analogous to Forward-Backward Algorithm for HMM
Backwards equations:

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

J_m is the **backwards Kalman gain matrix**
Initialized with μ_m and Σ_m from the Kalman Filter

Outline

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)

Baum-Welch-Algorithm

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

Baum-Welch-Algorithm

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

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 $\theta^* = \max_\theta p(X \mid \theta)$

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 $\theta = (A, B, \pi)$ with random initial conditions (if not no prior information is known)

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 $\theta = (A, B, \pi)$ with random initial conditions (if not no prior information is known)
- Use Forward an d Backwards Algorithms to calculate temporary variables:

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 $\theta = (A, B, \pi)$ with random initial conditions (if not no prior information is known)
- Use Forward an d Backwards Algorithms to calculate temporary variables: $\gamma_i(m) = p(z_m = i \mid x_{1:m}, \theta) = \frac{\alpha_i(m)\beta_i(m)}{\sum_j \alpha_j(m)\beta_j(m)}$

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 $\theta = (A, B, \pi)$ with random initial conditions (if not no prior information is known)
- Use Forward an d Backwards Algorithms to calculate temporary variables: $\gamma_i(m) = p(z_m = i \mid x_{1:m}, \theta) = \frac{\alpha_i(m)\beta_i(m)}{\sum_j \alpha_j(m)\beta_j(m)}$
-

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

$$a_{ij} = p(z_m = j \mid z_{m-1} = i), p(x_{m+1} \mid z_m = j) = b_j(x_{m+1})$$

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 $\theta = (A, B, \pi)$ with random initial conditions (if not no prior information is known)
- Use Forward an d Backwards Algorithms to calculate temporary variables: $\gamma_i(m) = p(z_m = i \mid x_{1:m}, \theta) = \frac{\alpha_i(m)\beta_i(m)}{\sum_j \alpha_j(m)\beta_j(m)}$
- $$\begin{aligned}\xi_{ij}(m) &= p(z_m = i, z_{m+1} = j \mid x_{1:m}, \theta)\\ &= \frac{\alpha_i(m) a_{ij} \beta_j(m+1) b_j(x_{m+1})}{\sum_m \alpha_m(m)\beta_m(m)}\end{aligned}$$

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

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

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

Further Readings

- [Mur12, chapter 17,18].

References

Kevin P. Murphy.
Machine learning: a probabilistic perspective.
The MIT Press, 2012.