Probabilistic Graphical Models

Lecture 7: Dynamic State Space Models

Volkan Cevher, Matthias Seeger
Ecole Polytechnique Fédérale de Lausanne

21/10/2011

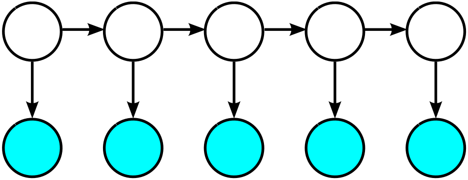
Announcement

Please bring your *Assignment 5* sheet along to next tutorial. Points were not recorded.

Outline

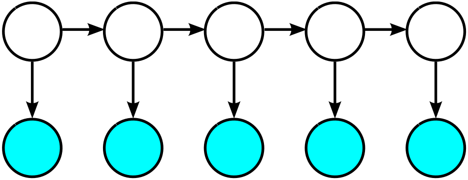
1. Hidden Markov Models
2. Linear Dynamical Systems
3. General Filtering / Smoothing

Forward in Time



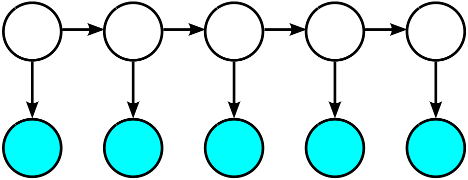
- Final lecture in part I: We'll do a big step . . . in time

Forward in Time



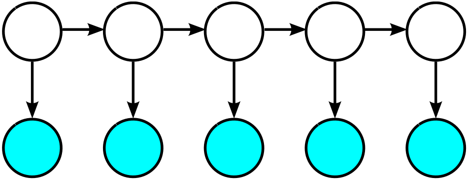
- Final lecture in part I: We'll do a big step . . . in time
- Why dynamic models?
  - Number one reason for causal dependence: Succession in time
  - Filtering, tracking, forward prediction, time series, sequential learning, . . .

Forward in Time



- Final lecture in part I: We'll do a big step . . . in time
- Why dynamic models?
  - Number one reason for causal dependence: Succession in time
  - Filtering, tracking, forward prediction, time series, sequential learning, . . .
- What's special about dynamic models?
  Only one direction (time arrow) $\rightarrow$ Linear (in)dependence $\rightarrow$ Markov chain $\rightarrow$ Chains are (simple) trees $\rightarrow$ Belief propagation!

Forward in Time



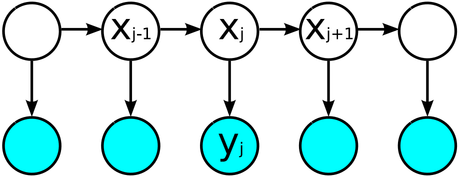
- Final lecture in part I: We'll do a big step . . . in time
- Why dynamic models?
  - Number one reason for causal dependence: Succession in time
  - Filtering, tracking, forward prediction, time series, sequential learning, . . .
- What's special about dynamic models?
  Only one direction (time arrow) $\rightarrow$ Linear (in)dependence $\rightarrow$ Markov chain $\rightarrow$ Chains are (simple) trees $\rightarrow$ Belief propagation!
- Markov chain: Present separates between past and future

$$(\boldsymbol{x}_{<i}) \perp (\boldsymbol{x}_{>i}) | \boldsymbol{x}_i$$

Vocabulary

Hidden Markov Model

F2



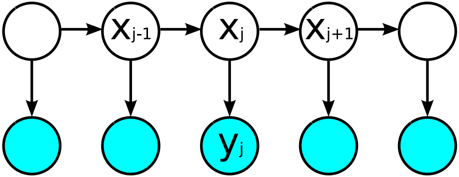
$P(\boldsymbol{y}_j \mid \boldsymbol{x}_j)$	Observation likelihood
$P(\boldsymbol{x}_j \mid \boldsymbol{x}_{j-1})$	Transition kernel
$P(\boldsymbol{x}_1)$	Initial state prior

Stationary Model: CPTs independent of j.
Notation: $\boldsymbol{x}_{<j} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_{j-1})$

Vocabulary

Hidden Markov Model

$P(\boldsymbol{y}_j \mid \boldsymbol{x}_j)$	Observation likelihood
$P(\boldsymbol{x}_j \mid \boldsymbol{x}_{j-1})$	Transition kernel
$P(\boldsymbol{x}_1)$	Initial state prior



Stationary Model: CPTs independent of j.

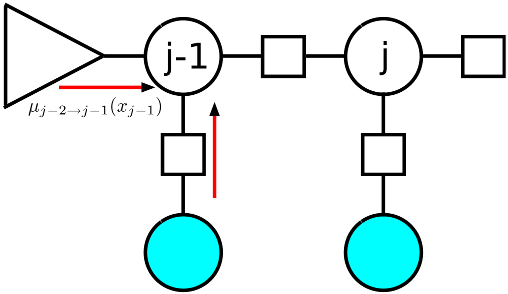
Notation: $\boldsymbol{x}_{<j} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_{j-1})$

- Filtering: $P(\boldsymbol{x}_j \mid \boldsymbol{y}_{\leq j})$ (sequential prediction)
- Smoothing: $P(\boldsymbol{x}_j \mid \boldsymbol{y}_{1\dots J})$ (inference given past and future)
- Learning: Fitting parameters of $P(\boldsymbol{y} \mid \boldsymbol{x})$, $P(\boldsymbol{x}_{\cdot} \mid \boldsymbol{x}_{\cdot-1})$:
  EM, based on smoothing [EM comes from HMM research]

Filtering: Information Forward Propagation

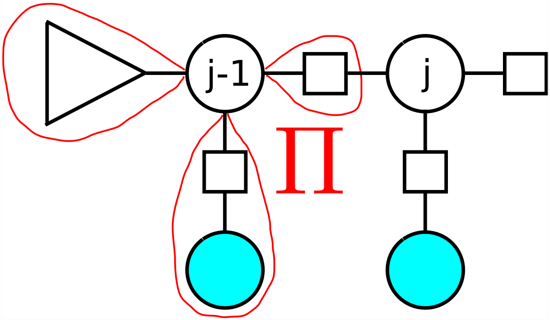
- Formal: Message passing on factor graph

  F3



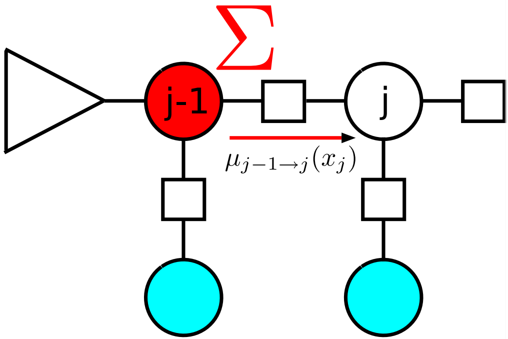
Filtering: Information Forward Propagation

- Formal: Message passing on factor graph



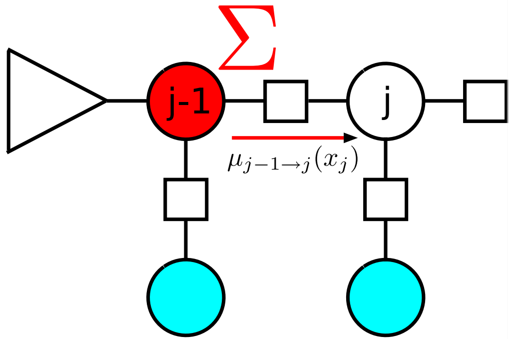
Filtering: Information Forward Propagation

- Formal: Message passing on factor graph



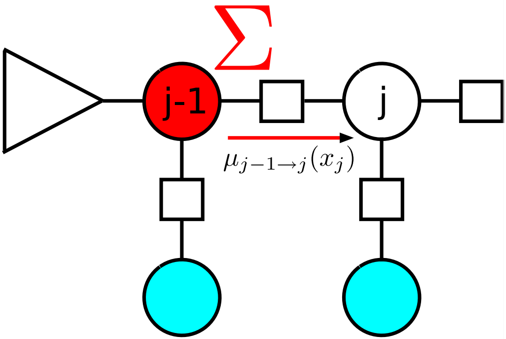
Filtering: Information Forward Propagation

- Formal: Message passing on factor graph
- Above formulae: Information forward propagation: $\mu_{(j-2)\to(j-1)}(x_{j-1})$: Prior "from the past"



Filtering: Information Forward Propagation

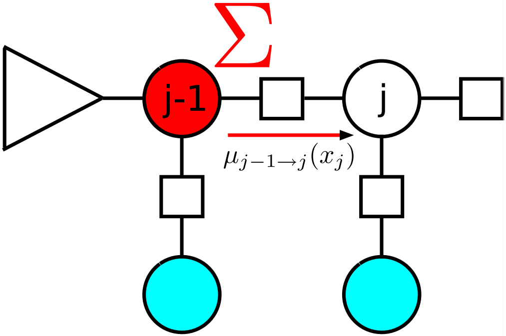
- Formal: Message passing on factor graph
- Above formulae: Information forward propagation: $\mu_{(j-2)\to(j-1)}(x_{j-1})$: Prior "from the past"



1. Measurement: Prior $\to$ posterior: $\propto \mu_{(j-2)\to(j-1)}(x_{j-1}) P(y_{j-1}|x_{j-1})$

Filtering: Information Forward Propagation

- Formal: Message passing on factor graph
- Above formulae: Information forward propagation: $\mu_{(j-2)\to(j-1)}(x_{j-1})$: Prior "from the past"

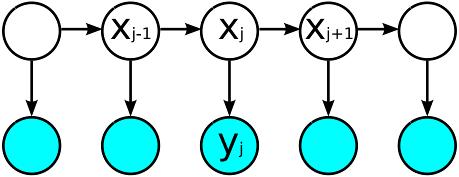


1. Measurement: Prior $\to$ posterior: $\propto \mu_{(j-2)\to(j-1)}(x_{j-1})P(y_{j-1}|x_{j-1})$
2. Diffusion, information propagation (marginalization): $\mu_{(j-1)\to j}(x_j) \propto \sum_{x_{j-1}}(\mu_{(j-2)\to(j-1)}(x_{j-1})P(y_{j-1}|x_{j-1}))P(x_j|x_{j-1})$

$\Rightarrow$ Know how to do (1), (2)? Can do BP!

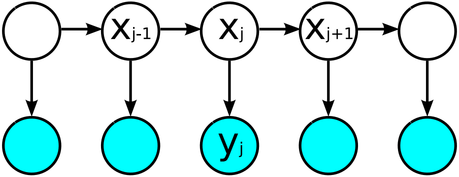
F3b

Sum-Product Algorithm for HMMs



- Backward messages: Exactly the same, just reverse time arrow
  1. Measurement: Prior $\rightarrow$ posterior: $\propto \mu_{j\leftarrow(j+1)}(x_j)P(y_j|x_j)$
  2. Diffusion, information propagation (marginalization):
     $\mu_{(j-1)\leftarrow j}(x_{j-1}) \propto \sum_{x_j} P(x_j|x_{j-1})(\mu_{j\leftarrow(j+1)}(x_j)P(y_j|x_j))$ F4

Sum-Product Algorithm for HMMs



- Backward messages: Exactly the same, just reverse time arrow
  1. Measurement: Prior $\rightarrow$ posterior: $\propto \mu_{j \leftarrow (j+1)}(x_j) P(y_j | x_j)$
  2. Diffusion, information propagation (marginalization): $\mu_{(j-1) \leftarrow j}(x_{j-1}) \propto \sum_{x_j} P(x_j | x_{j-1})(\mu_{j \leftarrow (j+1)}(x_j) P(y_j | x_j))$
- Forward / backward pass independent: can be run in parallel
- Posterior marginals:

F4b

$$
\begin{aligned}
P(x_j | \boldsymbol{y}_{1 \dots J}) &\propto \mu_{(j-1) \rightarrow j}(x_j) P(y_j | x_j) \mu_{j \leftarrow (j+1)}(x_j) \\
P(x_{j-1}, x_j | \boldsymbol{y}_{1 \dots J}) &\propto \mu_{(j-2) \rightarrow (j-1)}(x_{j-1}) P(y_{j-1} | x_{j-1}) P(x_j | x_{j-1}) \\
&\quad P(y_j | x_j) \mu_{j \leftarrow (j+1)}(x_j)
\end{aligned}
$$

HMMs in Practice

Enormously influential, both in practice and algorithm development

Speech recognition:
SR today: = HMMs with clever search tricks

- EM came from there (Baum, Welch: Forward-backward algorithm)
- Swiped field clean of anything else (rule-based, hand-coded, linguistic, ...) in 1970s. Early work at CMU (Baker, Lowerre) and IBM (Jelinek)
- x_j: Subphonemes. $\boldsymbol{y}_j$: Spectral features of acoustic waveform. $P(\boldsymbol{y}|x)$: Gaussian mixture
- One of the big success stories of statistical learning over other "loftier" approaches
- Today more industry than research: Big groups, big computers, huge amounts of data

HMMs in Practice

Enormously influential, both in practice and algorithm development

Bio-Informatics:
Introduced there early 90s by David Haussler (machine learning theorist, turned famous computational biologist)

- Before that: Dynamic programming sequence alignment (BLAST)
- Most macromolecules of organic chemistry are chains (some folded in complex ways):
  $x_j \in \{A, C, T, G\}$ (or triplets), $x_j \in \{\text{amino acids}\}$
- Sequence matching by pair HMMs (two $\boldsymbol{y}_j$ chains, common x_j)
- Gene finding: HGP estimates about $\#$ human genes: HMMs
- Protein categorization (homologues)
- Together with tree models: Phylogenetics, evolutionary history of species

Further Remarks

- How do I obtain $\mathrm{argmax}_{\{\boldsymbol{x}_j\}} \log P(\{\boldsymbol{y}_j\}, \{\boldsymbol{x}_j\})$?
  For example: How to I decode words from acoustic waveform?
  $\Rightarrow$ Max-product algorithm: Viterbi decoding
  - Learning with inner (Viterbi) maximization usually used in SR: Faster than EM (beam search, pruning)

Further Remarks

- How do I obtain $\operatorname{argmax}_{\{\boldsymbol{x}_j\}} \log P(\{\boldsymbol{y}_j\}, \{\boldsymbol{x}_j\})$?
  For example: How to I decode words from acoustic waveform?
  $\Rightarrow$ Max-product algorithm: Viterbi decoding
  - Learning with inner (Viterbi) maximization usually used in SR: Faster than EM (beam search, pruning)
- Simple modifications give nonstationary transition kernels (e.g., for non-geometric duration time)

Further Remarks

- How do I obtain $\mathrm{argmax}_{\{\boldsymbol{x}_j\}} \log P(\{\boldsymbol{y}_j\}, \{\boldsymbol{x}_j\})$?
  For example: How to I decode words from acoustic waveform?
  $\Rightarrow$ Max-product algorithm: Viterbi decoding
  - Learning with inner (Viterbi) maximization usually used in SR: Faster than EM (beam search, pruning)
- Simple modifications give nonstationary transition kernels (e.g., for non-geometric duration time)
- Not hard to do hierarchical / multi-scale HMMs: Nodes expand into HMMs themselves (of potentially different length)

Further Remarks

- How do I obtain $\operatorname{argmax}_{\{\boldsymbol{x}_j\}} \log P(\{\boldsymbol{y}_j\}, \{\boldsymbol{x}_j\})$?
  For example: How to I decode words from acoustic waveform?
  $\Rightarrow$ Max-product algorithm: Viterbi decoding
  - Learning with inner (Viterbi) maximization usually used in SR: Faster than EM (beam search, pruning)
- Simple modifications give nonstationary transition kernels (e.g., for non-geometric duration time)
- Not hard to do hierarchical / multi-scale HMMs: Nodes expand into HMMs themselves (of potentially different length)
- HMMs for large state spaces: Model structure within $\boldsymbol{x}_j$ itself
  - Factorial HMM [next lecture]

Further Remarks

- How do I obtain $\operatorname{argmax}_{\{\boldsymbol{x}_j\}} \log P(\{\boldsymbol{y}_j\}, \{\boldsymbol{x}_j\})$?
  For example: How to I decode words from acoustic waveform?
  $\Rightarrow$ Max-product algorithm: Viterbi decoding
  - Learning with inner (Viterbi) maximization usually used in SR: Faster than EM (beam search, pruning)
- Simple modifications give nonstationary transition kernels (e.g., for non-geometric duration time)
- Not hard to do hierarchical / multi-scale HMMs: Nodes expand into HMMs themselves (of potentially different length)
- HMMs for large state spaces: Model structure within $\boldsymbol{x}_j$ itself
  - Factorial HMM [next lecture]
- Common theme: On some level: Markov chain (usually latent). Belief propagation
  - Message sizes independent of sequence length
  - Running time linear in sequence length

Dynamical Systems

- The world is not discrete. Problems involving motion, co-ocurrence: Differential equations, continuous variables
- Reasoning about uncertainty in such problems: Dynamical state space models

Dynamical Systems

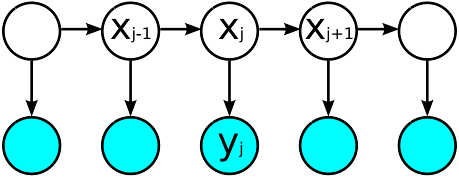
- The world is not discrete. Problems involving motion, co-ocurrence: Differential equations, continuous variables
- Reasoning about uncertainty in such problems: Dynamical state space models
- State: Finite set variables, containing all information to move on (separate past from future). Can include derivatives as well (location, orientation, velocity, angular velocity, acceleration, torque, . . .). Usually (partly) latent
- Reason about distribution over state, conditioned on past observations (filtering) or all observations (smoothing). Propagate such state distributions (belief states)

Dynamical Systems

- The world is not discrete. Problems involving motion, co-ocurrence: Differential equations, continuous variables
- Reasoning about uncertainty in such problems: Dynamical state space models
- State: Finite set variables, containing all information to move on (separate past from future). Can include derivatives as well (location, orientation, velocity, angular velocity, acceleration, torque, ...). Usually (partly) latent
- Reason about distribution over state, conditioned on past observations (filtering) or all observations (smoothing). Propagate such state distributions (belief states)
- Use this inference for higher order tasks
  - Learning about environment, world model, sensor accuracy
  - Planning behaviour, interaction which modifies environment

$\Rightarrow$ Whatever you do: Inference is at the bottom

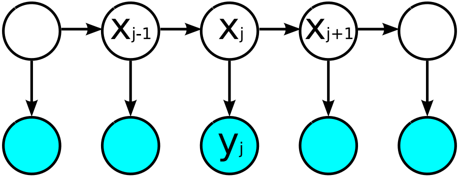
Linear Dynamical System



- Example: Moving robot localization

F8

Linear Dynamical System

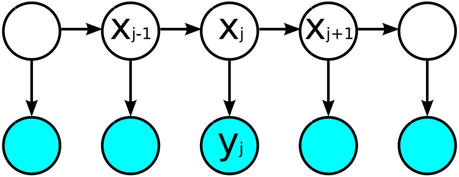


- Example: Moving robot localization
- Local conditional probabilities: Linear-Gaussian

$$\begin{array}{lll} \boldsymbol{x}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{G}_t \varepsilon_{1,t}, & \varepsilon_{1,t} \sim N(\boldsymbol{0}, \boldsymbol{I}) & \text{Transition prior} \\ \boldsymbol{y}_t = \boldsymbol{C}_t \boldsymbol{x}_t + \varepsilon_{2,t}, & \varepsilon_{2,t} \sim N(\boldsymbol{0}, \boldsymbol{\Psi}_t) & \text{Observation likelihood} \end{array}$$

$\varepsilon_{k,t}$ independent of others.
Stationary LDS: $\boldsymbol{A}$, $\boldsymbol{G}$, $\boldsymbol{C}$, $\boldsymbol{\Psi}$ independent of t

Linear Dynamical System



- Example: Moving robot localization
- Local conditional probabilities: Linear-Gaussian

$$\begin{array}{lll} \boldsymbol{x}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{G}_t \varepsilon_{1,t}, & \varepsilon_{1,t} \sim N(\boldsymbol{0}, \boldsymbol{I}) & \text{Transition prior} \\ \boldsymbol{y}_t = \boldsymbol{C}_t \boldsymbol{x}_t + \varepsilon_{2,t}, & \varepsilon_{2,t} \sim N(\boldsymbol{0}, \boldsymbol{\Psi}_t) & \text{Observation likelihood} \end{array}$$

  $\varepsilon_{k,t}$ independent of others.
  Stationary LDS: $\boldsymbol{A}$, $\boldsymbol{G}$, $\boldsymbol{C}$, $\boldsymbol{\Psi}$ independent of t
- Inference in such a model? Combine what you know:
  - HMM for discrete latent states
  - Factor analysis for linear-Gaussian model, independent states

Behind the Equations

- Behind the equation mess in engineering textbooks, there is a simple idea. What you need to remember:
  - Above formulae: That idea, and the generic primitives it requires
  - Below formulae: Numerically uncritical way of implementation

Behind the Equations

- Behind the equation mess in engineering textbooks, there is a simple idea. What you need to remember:
  - Above formulae: That idea, and the generic primitives it requires
  - Below formulae: Numerically uncritical way of implementation
- Simple idea:
  - All distributions / messages in this model: Gaussian.
    You need to maintain / pass:
    - Moment parameters (mean, covariance), or
    - Natural parameters
  - Belief propagation primitives: Measurement (prior $\rightarrow$ posterior), information propagation (marginalization)

Behind the Equations

- Behind the equation mess in engineering textbooks, there is a simple idea. What you need to remember:
  - Above formulae: That idea, and the generic primitives it requires
  - Below formulae: Numerically uncritical way of implementation
- Simple idea:
  - All distributions / messages in this model: Gaussian.
    You need to maintain / pass:
    - Moment parameters (mean, covariance), or
    - Natural parameters
  - Belief propagation primitives: Measurement (prior $\rightarrow$ posterior), information propagation (marginalization)
- Below formulae: Get these primitives right
  - Use linear algebra to get them in the right form
  - Use numerically trusted solutions for elementary steps

Kalman Filtering

Filtering with moment parameters:

$$N(\boldsymbol{x}_{t-1}|\boldsymbol{\mu}_{t-1|t-1}, \boldsymbol{\Sigma}_{t-1|t-1}) \xrightarrow{\text{info. prop.}} N(\boldsymbol{x}_t|\boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})$$
$$\xrightarrow{\text{measurement}} N(\boldsymbol{x}_t|\boldsymbol{\mu}_{t|t}, \boldsymbol{\Sigma}_{t|t})$$

- Information propagation: $\boldsymbol{x}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{G}_t \varepsilon_{1,t}$

Kalman Filtering

Filtering with moment parameters:

$$N(\boldsymbol{x}_{t-1}|\boldsymbol{\mu}_{t-1|t-1}, \Sigma_{t-1|t-1}) \stackrel{\text{info. prop.}}{\longrightarrow} N(\boldsymbol{x}_t|\boldsymbol{\mu}_{t|t-1}, \Sigma_{t|t-1})$$
$$\stackrel{\text{measurement}}{\longrightarrow} N(\boldsymbol{x}_t|\boldsymbol{\mu}_{t|t}, \Sigma_{t|t})$$

- Information propagation: $\boldsymbol{x}_t = \boldsymbol{A}_t\boldsymbol{x}_{t-1} + \boldsymbol{G}_t\varepsilon_{1,t}$

$$\boldsymbol{\mu}_{t|t-1} = \boldsymbol{A}_t\boldsymbol{\mu}_{t-1|t-1}, \quad \Sigma_{t|t-1} = \boldsymbol{A}_t\Sigma_{t-1|t-1}\boldsymbol{A}_t^T + \boldsymbol{G}_t\boldsymbol{G}_t^T$$

- Measurement: “Prior” $N(\boldsymbol{x}_t|\boldsymbol{\mu}_{t|t-1}, \Sigma_{t|t-1})$. Likelihood $N(\boldsymbol{y}_t|\boldsymbol{C}_t\boldsymbol{x}_t, \Psi_t)$. Posterior?

F10

Kalman Filtering

Filtering with moment parameters:

$$N(\boldsymbol{x}_{t-1}|\boldsymbol{\mu}_{t-1|t-1}, \Sigma_{t-1|t-1}) \overset{\text{info. prop.}}{\longrightarrow} N(\boldsymbol{x}_t|\boldsymbol{\mu}_{t|t-1}, \Sigma_{t|t-1})$$

$$\overset{\text{measurement}}{\longrightarrow} N(\boldsymbol{x}_t|\boldsymbol{\mu}_{t|t}, \Sigma_{t|t})$$

- Information propagation: $\boldsymbol{x}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{G}_t \varepsilon_{1,t}$

$$\boldsymbol{\mu}_{t|t-1} = \boldsymbol{A}_t \boldsymbol{\mu}_{t-1|t-1}, \quad \Sigma_{t|t-1} = \boldsymbol{A}_t \Sigma_{t-1|t-1} \boldsymbol{A}_t^T + \boldsymbol{G}_t \boldsymbol{G}_t^T$$

- Measurement: "Prior" $N(\boldsymbol{x}_t|\boldsymbol{\mu}_{t|t-1}, \Sigma_{t|t-1})$. Likelihood $N(\boldsymbol{y}_t|\boldsymbol{C}_t \boldsymbol{x}_t, \Psi_t)$. Posterior?

$$\Sigma_{t|t} = \mathrm{Cov}[(\boldsymbol{x}_t\, \boldsymbol{y}_t)]/\mathrm{Cov}[\boldsymbol{y}_t] = \Sigma_{t|t-1} - \Sigma_{t|t-1} \boldsymbol{C}_t^T \boldsymbol{E}_t^{-1} \boldsymbol{C}_t \Sigma_{t|t-1}$$

$$\boldsymbol{\mu}_{t|t} = \mathrm{E}[\boldsymbol{x}_t] + \mathrm{Cov}[\boldsymbol{x}_t, \boldsymbol{y}_t]\mathrm{Cov}[\boldsymbol{y}_t]^{-1}(\boldsymbol{y}_t - \mathrm{E}[\boldsymbol{y}_t])$$

$$= \boldsymbol{\mu}_{t|t-1} + \underbrace{\Sigma_{t|t-1} \boldsymbol{C}_t^T \boldsymbol{E}_t^{-1}}_{\text{Kalman gain}}(\boldsymbol{y}_t - \boldsymbol{C}_t \boldsymbol{\mu}_{t|t-1}), \quad \boldsymbol{E}_t = \Psi_t + \boldsymbol{C}_t \Sigma_{t|t-1} \boldsymbol{C}_t^T$$

Remarks

- Kalman gain matrix:

$$\boldsymbol{K}_t = \Sigma_{t|t-1}\boldsymbol{C}_t^T(\Psi_t + \boldsymbol{C}_t\Sigma_{t|t-1}\boldsymbol{C}_t^T)^{-1}\left\{= \mathrm{Cov}[\boldsymbol{x}_t, \boldsymbol{y}_t]\mathrm{Cov}[\boldsymbol{y}_t]^{-1}\right\}$$

  Residual error $\boldsymbol{y}_t - \mathrm{E}[\boldsymbol{y}_t|\boldsymbol{y}_{<t}] \rightarrow$ correction mean estimate.
  Can also write: $\Sigma_{t|t} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{C}_t)\Sigma_{t|t-1}$

Remarks

- Kalman gain matrix:

$$\boldsymbol{K}_t = \Sigma_{t|t-1}\boldsymbol{C}_t^T(\Psi_t + \boldsymbol{C}_t\Sigma_{t|t-1}\boldsymbol{C}_t^T)^{-1} \left\{= \mathrm{Cov}[\boldsymbol{x}_t, \boldsymbol{y}_t]\mathrm{Cov}[\boldsymbol{y}_t]^{-1}\right\}$$

  Residual error $\boldsymbol{y}_t - \mathrm{E}[\boldsymbol{y}_t|\boldsymbol{y}_{<t}] \to$ correction mean estimate.
  Can also write: $\Sigma_{t|t} = (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{C}_t)\Sigma_{t|t-1}$

- Recall: In the moment parameterization:

Information propagation	Simple
Measurement	Difficult (needs matrix factorization)

  In the natural parameterization, these roles are reversed

- Information filter: Propagate natural parameters $\boldsymbol{r}_{t|t}$, $\boldsymbol{S}_{t|t}$ instead of moment parameters $[\boldsymbol{S}_{t|t} = \Sigma_{t|t}^{-1}, \boldsymbol{r}_{t|t} = \Sigma_{t|t}^{-1}\boldsymbol{\mu}_{t|t}]$

In Practice

"Below the formulae": What's he talking about?

$\Rightarrow$ In practice, neither of them work (on real problems)

- In theory, $\Sigma_{t|t}$ or $\boldsymbol{S}_{t|t}$ stay positive definite. In practice they don't!
- Root of problem: Information propagation / measurement simple linear in different parameterizations. Conversion (matrix inversion) prone to numerical errors

In Practice

"Below the formulae": What's he talking about?
$\Rightarrow$ In practice, neither of them work (on real problems)

- In theory, $\Sigma_{t|t}$ or $\boldsymbol{S}_{t|t}$ stay positive definite. In practice they don't!
- Root of problem: Information propagation / measurement simple linear in different parameterizations. Conversion (matrix inversion) prone to numerical errors
- First improvement: Propagate matrix factorization:

  Kalman square root filter $\Sigma_{t|t} = \boldsymbol{F}_{t|t}\boldsymbol{F}_{t|t}^T$. Propagate $\boldsymbol{F}_{t|t}$
  Information square root filter $\Sigma_{t|t}^{-1} = \boldsymbol{F}_{t|t}\boldsymbol{F}_{t|t}^T$. Propagate $\boldsymbol{F}_{t|t}$
- Further improvements: Formulate as weighted least squares problem. Use stable LS method from numerical mathematics

  [Paige, Saunders: Least Squares Estimation of Discrete Linear Dynamic Systems Using Orthogonal Transformations (1977)]

Smoothing

- All approaches use filtering for forward pass
- Rauch-Tung-Striebel (RTS) smoother: Backward pass computes marginals $\mathrm{E}[\boldsymbol{x}_t|D]$, $\mathrm{Cov}[\boldsymbol{x}_t|D]$ directly, $D = \{\boldsymbol{y}_t\}$. Idea:

$$
\begin{aligned}
P(\boldsymbol{x}_t|D) &= \int P(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, D) P(\boldsymbol{x}_{t+1}|D)\, d\boldsymbol{x}_{t+1} \\
&\stackrel{!}{=} \int P(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{y}_{\le t}) P(\boldsymbol{x}_{t+1}|D)\, d\boldsymbol{x}_{t+1} \quad [\boldsymbol{x}_t \perp \boldsymbol{y}_{>t} | \boldsymbol{x}_{t+1}]
\end{aligned}
$$

Work out moments of $P(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{y}_{\le t})$ from filtering variables. Average $\boldsymbol{x}_{t+1}$ over $P(\boldsymbol{x}_{t+1}|D)$. Details: In your exercises

Smoothing

- All approaches use filtering for forward pass
- Rauch-Tung-Striebel (RTS) smoother: Backward pass computes marginals $\mathrm{E}[\boldsymbol{x}_t|D]$, $\mathrm{Cov}[\boldsymbol{x}_t|D]$ directly, $D = \{\boldsymbol{y}_t\}$. Idea:

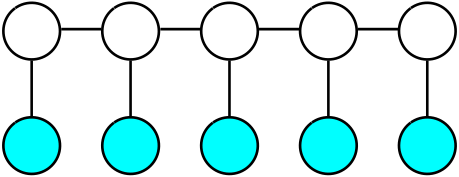
$$\begin{aligned} P(\boldsymbol{x}_t|D) &= \int P(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, D) P(\boldsymbol{x}_{t+1}|D)\, d\boldsymbol{x}_{t+1} \\ &\stackrel{!}{=} \int P(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{y}_{\leq t}) P(\boldsymbol{x}_{t+1}|D)\, d\boldsymbol{x}_{t+1} \quad [\boldsymbol{x}_t \perp \boldsymbol{y}_{>t}|\boldsymbol{x}_{t+1}] \end{aligned}$$

  Work out moments of $P(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{y}_{\leq t})$ from filtering variables. Average $\boldsymbol{x}_{t+1}$ over $P(\boldsymbol{x}_{t+1}|D)$. Details: In your exercises
- Two-filter smoothing: Analogous to forward-backward BP
  - Run backward filter (in parallel to forward filter) F13
  - Combine results by Gaussian product formula. Do not count observation twice!

Learning

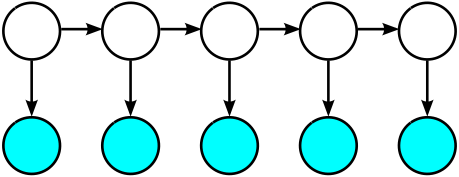
- Recall last lecture: Most difficult part of learning is inference
- EM algorithm: E step is smoothing. M step: Like in factor analysis
- Gradient-based optimization: Average gradients of log-potentials over marginal posterior (smoothing)
- Formulae even worse, but we are not impressed
  - Above formulae: Decomposition:
    - Marginal inference (smoothing)
    - Gradient accumulation, given marginals
    - Parameter updates
  - Below formulae: Use stable filtering / smoothing implementation. Gradient accumulation typically harmless

Conditional Random Fields



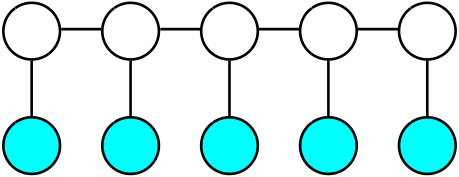
- Undirected sequence model. Different properties through global normalization (HMM: local normalization)
- Markov random field with tree graph
- Heavily used in text / language modelling (labeling, named entity recognition)
- Training with complete data (all $\boldsymbol{x}_t$ given): Iterative, but convex optimization (for log-linear potentials). Can be done very efficiently (Quasi Newton optimization; approximate Newton optimization with Hessian-vector product)
- Traning with incomplete data: EM outer loop required

Conditional Random Fields



- Undirected sequence model. Different properties through global normalization (HMM: local normalization)
- Markov random field with tree graph
- Heavily used in text / language modelling (labeling, named entity recognition)
- Training with complete data (all $\boldsymbol{x}_t$ given): Iterative, but convex optimization (for log-linear potentials). Can be done very efficiently (Quasi Newton optimization; approximate Newton optimization with Hessian-vector product)
- Traning with incomplete data: EM outer loop required

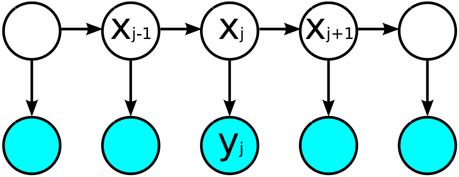
Conditional Random Fields



- Undirected sequence model. Different properties through global normalization (HMM: local normalization)
- Markov random field with tree graph
- Heavily used in text / language modelling (labeling, named entity recognition)
- Training with complete data (all $\boldsymbol{x}_t$ given): Iterative, but convex optimization (for log-linear potentials). Can be done very efficiently (Quasi Newton optimization; approximate Newton optimization with Hessian-vector product)
- Traning with incomplete data: EM outer loop required

General State Space Models

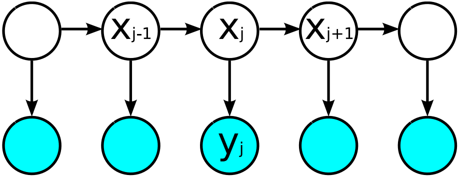
- We've had state space models
  - All discrete: Multinomial family
  - All linear-Gaussian: Gaussian family



- What about other situations?
  - Nonlinear dynamical system: Transition / observation mapping nonlinear. Noise dependent on current state
  - Switching state space model: State consists of continuous and discrete variables

$\Rightarrow$ Graph is still a chain. Efficient inference by BP?

General State Space Models



- We've had state space models
  - All discrete: Multinomial family
  - All linear-Gaussian: Gaussian family
- What about other situations?
  - Nonlinear dynamical system: Transition / observation mapping nonlinear. Noise dependent on current state
  - Switching state space model: State consists of continuous and discrete variables

$\Rightarrow$ Graph is still a chain. Efficient inference by BP?

- Problem: Only multinomial, Gaussian families
  - Closed under conditioning and marginalization
  - Fixed-size parameterization

$\Rightarrow$ Inference for general state space models: NP hard

F16

Approximate Filtering

Dynamic programming requires fixed-size message representations
$\Rightarrow$ Blow-up has to be countered by approximations

- Approximate transition / observation potentials locally
  - Extended Kalman filter: Linearize transition / observation mapping by Taylor expansion at $\boldsymbol{\mu}_{t-1|t-1}$ / $\boldsymbol{\mu}_{t|t-1}$ resp. Use exact propagation with approximated potentials

Approximate Filtering

Dynamic programming requires fixed-size message representations
$\Rightarrow$ Blow-up has to be countered by approximations

- Approximate transition / observation potentials locally
  - Extended Kalman filter: Linearize transition / observation mapping by Taylor expansion at $\boldsymbol{\mu}_{t-1|t-1}$ / $\boldsymbol{\mu}_{t|t-1}$ resp. Use exact propagation with approximated potentials
- Use exact local propagation, project resulting message back into representation family. Projection done optimally by matching family moments (mean, covariance for Gaussian). Variants:
  - Assumed density filter
  - Unscented filter (cheap quadrature by exact monomials)

Approximate Filtering

Dynamic programming requires fixed-size message representations
$\Rightarrow$ Blow-up has to be countered by approximations

- Approximate transition / observation potentials locally
  - Extended Kalman filter: Linearize transition / observation mapping by Taylor expansion at $\boldsymbol{\mu}_{t-1|t-1}$ / $\boldsymbol{\mu}_{t|t-1}$ resp. Use exact propagation with approximated potentials
- Use exact local propagation, project resulting message back into representation family. Projection done optimally by matching family moments (mean, covariance for Gaussian). Variants:
  - Assumed density filter
  - Unscented filter (cheap quadrature by exact monomials)
- EKF cheap and cheerful. ADF works better in general, but needs quadrature to approximate moments. Projection more general:
  - EKF can be seen as special backprojection as well

What about Smoothing?

- Approximate filters: One-shot approach, local approximations never re-visited. But: Approximate inference is iterative
- Correct learning requires marginals given all data (also future) $\Rightarrow$ Smoothing

What about Smoothing?

- Approximate filters: One-shot approach, local approximations never re-visited. But: Approximate inference is iterative
- Correct learning requires marginals given all data (also future) $\Rightarrow$ Smoothing
- Options:
  - Apply filter approximation technique to smoother (e.g., two-filter smoother) $\Rightarrow$ not iterative
  - Better: Use principled approximate inference framework [expectation propagation, part II]
- General warning: Numerically even more difficult than inference in fixed LDS $\Rightarrow$ Attention to numerical details essential

Wrap-Up

- Hidden Markov model (discrete states): Non-Markovian behaviour from Markovian ingredients
- Linear dynamical system: Simple idea, messy equations. Does not work without numerically careful implementation
- Conditional random field: Alternative to HMM for large text / language problems
- Filtering / smoothing for general state space models: Approximation by (moment matching) backprojection