UNIVERSITÄT OSNABRÜCK

University of Osnabrück, Institute of Cognitive Science

Machine Learning

12 – Temporal Probability Models

SS 2018

Gunther Heidemann

Overview

Temporal probability models

- Modeling uncertainty for temporal processes
- Markov processes
- Temporal inference:
  - Filtering
  - Prediction
  - Smoothing
  - Most likely explanation (Viterbi algorithm)
- Brief overview of dynamic Bayes Networks (DBN)

Textbook:

Stuart Russell, Peter Norvig: *Artificial Intelligence*, Pearson

Time and uncertainty

Aim: Track and predict processes over time

Examples: Diabetes management; motor management

Description of a process:

- Description by discrete states (using discrete time t).
- State at t is specified by a set X_t of ***unobservable variables*** (= this is the problem!), e.g.,

$$X_t = \{BloodSugar_t, StomachContents_t\}.$$

- The state becomes visible only by a set E_t of observable evidence variables, e.g.,

$$E_t = \{MeasuredBloodSugar_t, FoodEaten_t , PulseRate_t\}.$$

Notation for a span of time:

$$X_{a:b} = X_a, X_{a+1}, \ldots , X_{b-1}, X_b.$$

Markov processes

Markov assumption for a process described by variable X_t (discrete time):

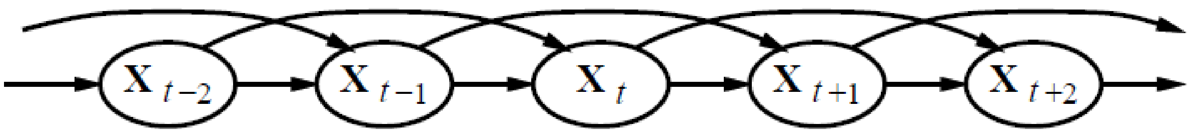
> X_t depends only on a *bounded subset* of the variables $X_{0:t-1}$.

First order Markov process: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-1})$

Second order Markov process: $P(X_t \mid X_{0:t-1}) = P(X_t \mid X_{t-2}, X_{t-1})$

First order

Second order



[RN]

Markov processes

Real world:

- Assumption of a first order Markov process simplifies modelling,
- but does usually not strictly hold.

Improvements:

1. Assume Markov process of higher order.
2. Augment knowledge about the state by additional evidence variables.

Example: Moving robot

Augment state description *(position$_t$, velocity$_t$)* by *battery$_t$*.

Markov sensor assumption

Markov assumption for sensors:

Sensor output depends only on the current value of the evidence variable observed by the sensor:

$$P(E_t \mid X_{0:t}, E_{0:t-1}) = P(E_t \mid X_t)$$

Example: Speedometer

Observed velocity E_t depends only on current velocity X_t, not on past values X_{t-n}.

Counter example: Water gauge for hydroponics.

- Unobservable variable: Height of water X_t
- Observed evidence variable: Measured height E_t
- If X exceeds a maximum height for weeks, algae block the water gauge → no Markov sensor.

Stationary Markov processes

Stationary process:

The world is changing, but the rules underlying both this change and its observation remain the same.

→ $P(X_t \mid X_{0:t-1}, t) = P(X_t \mid X_{0:t-1})$.

Together with the Markov assumptions for the process and the sensor, all we need to describe the measurements are the

transition model $P(X_t \mid X_{t-1})$ and the

sensor model (model of observation) $P(E_t \mid X_t)$,

which both remain fixed for all t.

Stationary Markov processes

Example of a stationary state transition model:

- X is water level of hydroponics.
- Add approx. 1l water every week (Gaussian distribution with mean = 1l).

Counter example:

- In summer, the weather becomes hot and the average is increased to 1,2l (rules of the world have changed).

Example for a stationary sensor model:

- Weight measurement using spring:
- X is the weight, E the weight measurement (which may exhibit a systematic but stationary error).

Counter example:

- Over time, the spring looses strength.

Inference

Problem:

- Stationary first order Markov process with known state transition model $\mathbf{P}(X_t \mid X_{t-1})$.
- Markov sensor with known sensor model $\mathbf{P}(E_t \mid X_t)$.

Inference:

Several types of inference which are different mixtures of two basic problems:

- Infer a past or the current value of an unobservable state variable X from the observable evidence variables.
- Prediction.

Inference

The type of inference depends on the times for which the probability distribution of X is inferred from past to current evidence values e.

Let T denote the current time (“now”):

1. Filtering: $\mathbf{P}(X_T \mid e_{1:T})$

   Infer probability distribution $\mathbf{P}$ of current state X_T from current evidence value e_T and past evidence values $e_{1:T-1}$ (but note $x_1 \ldots x_{T-1}$ are unknown).

   $\mathbf{P}(X_T \mid e_{1:T})$ is called a *belief state*. This is the input for the decision process of a rational agent.

2. Prediction: $\mathbf{P}(X_{T+K} \mid e_{1:T})$, $K > 0$

   Predict future value of X_{T+K} from evidence values $e_{1:T}$. Same as filtering, but the future evidence values $e_{T+1:T+K}$ are unknown.

Inference

3. Smoothing: $\mathbf{P}(X_K \mid e_{1:T})$, $0 < K < T$

   Infer the probability distribution of X at the past time K from earlier evidence values $e_{1:K-1}$ and later evidence values $e_{K+1:T}$.

   Yields better estimate for past states then filtering.

4. Most likely explanation: $\arg\max_{X_{1:T}} \mathbf{P}(X_{1:T} \mid e_{1:T})$

   Infer all $X_{1:T}$ from all evidence values $e_{1:T}$.

   Example: Speech recognition.

Inference: Filtering

Given: Transition model $\mathbf{P}(X_t \mid X_{t-1})$ and sensor model $\mathbf{P}(E_t \mid X_t)$.

Wanted: Probabilities for X_T from $e_{1:T}$ (better than mere estimation of X_T from e_T).

Principle: Recursive state estimation algorithm which starts with an assumption for X_0 and can infer values at $t+1$ from t.

For $t = 0, 1, \ldots T-1$:

$$
\begin{aligned}
\mathbf{P}(X_{t+1} \mid e_{1:t+1}) \\
&= \mathbf{P}(X_{t+1} \mid e_{1:t}, e_{t+1}) \\
&= \alpha \mathbf{P}(e_{t+1} \mid X_{t+1}, e_{1:t})\ \mathbf{P}(X_{t+1} \mid e_{1:t}) && \text{(Bayes)} \\
&= \alpha \mathbf{P}(e_{t+1} \mid X_{t+1})\ \mathbf{P}(X_{t+1} \mid e_{1:t}) && \text{(Markov sensor)} \\
&= \alpha \mathbf{P}(e_{t+1} \mid X_{t+1}) \sum_{x_t} \mathbf{P}(X_{t+1} \mid x_t, e_{1:t})\ P(x_t \mid e_{1:t}) && \text{(summing out } X_t\text{)} \\
&= \alpha\, \mathbf{P}(e_{t+1} \mid X_{t+1}) \sum_{x_t} \mathbf{P}(X_{t+1} \mid x_t)\ P(x_t \mid e_{1:t}) && \text{(Markov process)} \\
&= \alpha\ \textit{sensor model}_{t+1} \sum_{x_t} \textit{transition model}_{t+1,t}\ \textit{probability distribution}_t
\end{aligned}
$$

Filtering

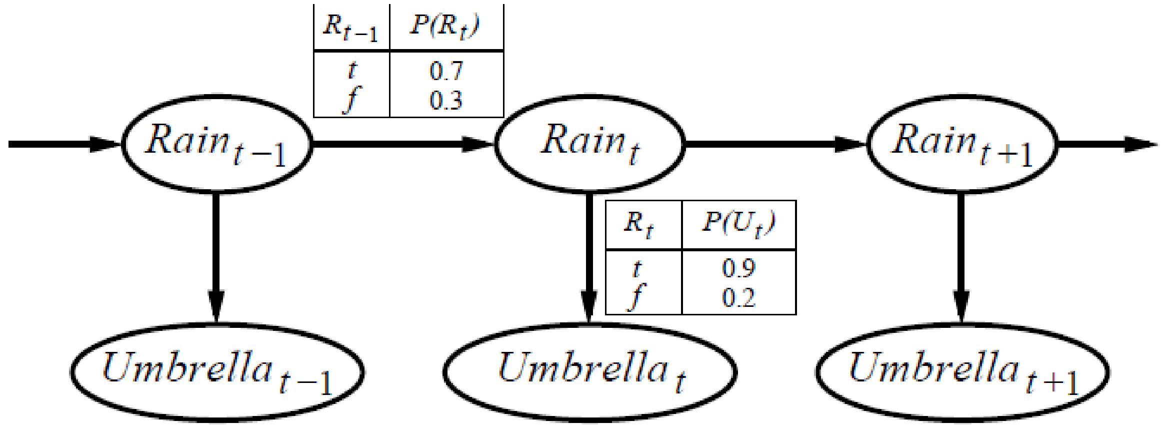
Structure of recursion:

$$f_{1:t+1} = \quad Forward(f_{1:t}, e_{t+1}) \quad \text{with} \quad f_{1:t} = \mathbf{P}(X_t \mid e_{1:t})$$

Requires constant time and memory for each step, independent of t.

Example [RN]:

- Living in a bunker, time steps are days.
- Only your boss is allowed to go outside.
- You can infer whether it is raining (X_t) only from his umbrella.



R_{t-1}	$P(R_t)$
t	0.7
f	0.3

R_t	$P(U_t)$
t	0.9
f	0.2

[RN]

Filtering

Transition model:

$\mathbf{P}(X_t \mid X_t{-}1) = \mathbf{P}(Rain_t \mid Rain_{t-1})$ with $\mathbf{P}(X_t \mid X_{t-1} = true) = \langle 0.7, 0.3 \rangle$

$\mathbf{P}(X_t \mid X_{t-1} = false) = \langle 0.3, 0.7 \rangle$

Sensor model:

$\mathbf{P}(E_t \mid X_t) = \mathbf{P}(Umbrella_t \mid Rain_t)$ with $\mathbf{P}(E_t \mid X_t = true) = \langle 0.9, 0.1 \rangle$

$\mathbf{P}(E_t \mid X_t = false) = \langle 0.2, 0.8 \rangle$

What is the probability for rain $P(X_2)$ on the second day (T=2), if

- Day 1: Umbrella $e_1 = true$.
- Day 2: Umbrella $e_2 = true$.
- We need an assumption about the probability of rain for day 0:

$$\mathbf{P}(X_0) = \langle 0.5, 0.5 \rangle$$

Filtering

$$\mathbf{P}(X_{t+1}|e_{1:t+1}) = \alpha\, \mathbf{P}(e_{t+1} \mid X_{t+1}) \quad \Sigma_{x_t}\ \mathbf{P}(X_{t+1} \mid x_t)\, P(x_t \mid e_{1:t})$$

$$\mathbf{P}(X_2 \mid e_{1:2}) = \alpha\, \mathbf{P}(e_2 \mid X_2) \quad \Sigma_{x_1}\ \mathbf{P}(X_2 \mid x_1)\ P(x_1 \mid e_1)$$

$$= \alpha\, \mathbf{P}(e_2 \mid X_2) \quad [\, \mathbf{P}(X_2 \mid x_1{=}t)\ 0.818 + \mathbf{P}(X_2 \mid x_1{=}f)\ 0.182]$$

$$= \alpha <0.9, 0.2> \ [<0.7, 0.3>\ 0.818 + <0.3, 0.7>\ 0.182]$$

$$= \alpha <0.9, 0.2> \ <0.627, 0.373> = \alpha <0.565, 0.075>$$

$$= <0.883, 0.117>$$

$$\mathbf{P}(X_1 \mid e_1)$$

$$= \alpha'\, \mathbf{P}(e_1 \mid X_1) \quad \Sigma_{x_0} \mathbf{P}(X_1 \mid x_0)\ P(x_0)$$

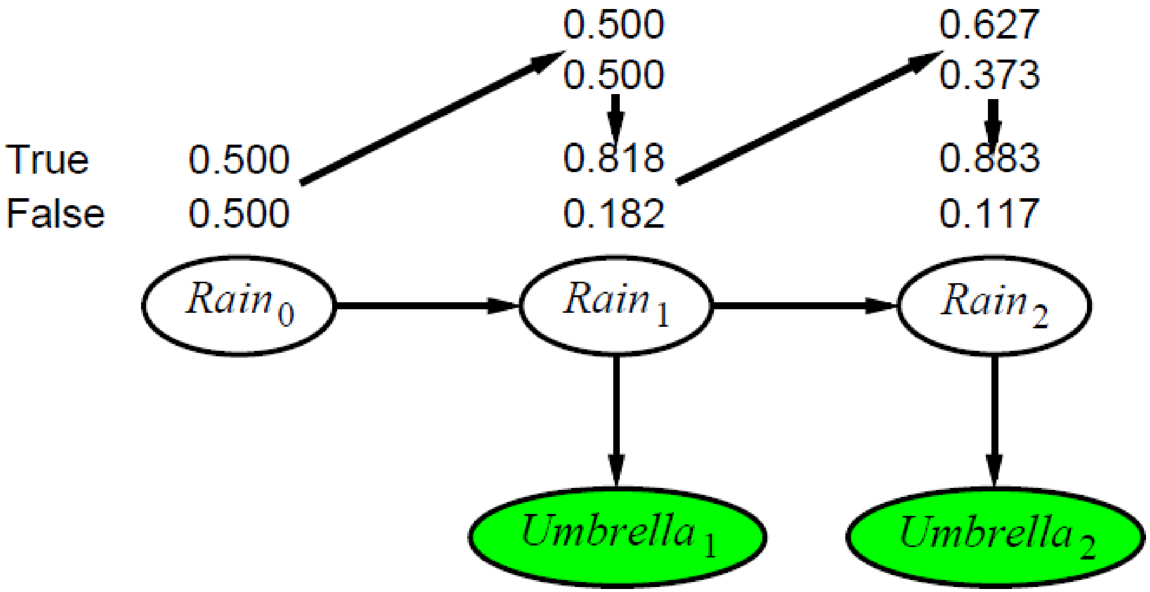
$$= \alpha'\, \mathbf{P}(e_1 \mid X_1)\ [\, \mathbf{P}(X_1 \mid x_0{=}t)\, P(x_0{=}t) + \mathbf{P}(X_1 \mid x_0{=}f)\, P(x_0{=}f)\ ]$$

$$= \alpha'\, \mathbf{P}(e_1 \mid X_1)\ [<0.7, 0.3>\ 0.5 + <0.3, 0.7>\ 0.5]$$

$$= \alpha'\, \mathbf{P}(e_1 \mid X_1) <0.5, 0.5>$$

$$= \alpha' <0.9, 0.2> \ <0.5, 0.5> = \alpha' <0.45, 0.1> = <0.818, 0.182>$$

Filtering



[RN]

Prediction

Given: Transition model $\mathbf{P}(X_t \mid X_{t-1})$ and sensor model $\mathbf{P}(E_t \mid X_t)$.

Wanted: X_{T+K} from $e_{1:T}$.

Idea:

- Filtering up to T
- After T: K further steps without new evidences (we lack $e_{T+1:T+K}$).
- New recursion: $T+k \rightarrow T+k+1$

For $k = 0, 1, \ldots K-1$:

$$\mathbf{P}(X_{T+k+1} \mid e_{1:T}) = \Sigma_{x_{T+k}} \mathbf{P}(X_{T+k+1} \mid x_{T+k}) \, P(x_{T+k} \mid e_{1:T})$$

$$= \Sigma_{x_{T+k}} \; \mathit{transition\ model}_{T+k+1,T+k} \; \mathit{distribution}_{T+k}$$

The farther we predict the future without new evidences, the more the distribution is dominated by the transition model.

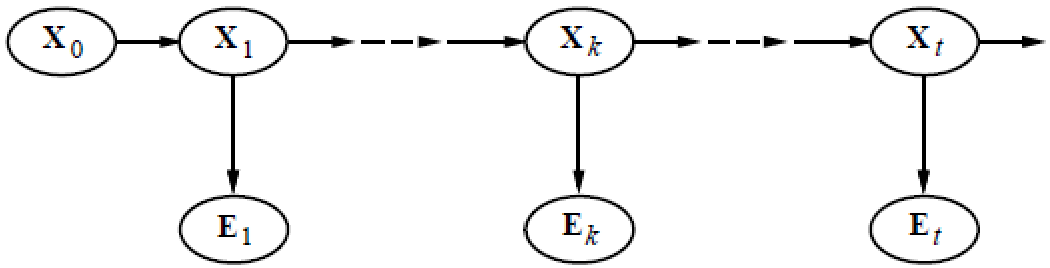
Smoothing

Given: Transition model $\mathbf{P}(X_t \mid X_{t-1})$ and sensor model $\mathbf{P}(E_t \mid X_t)$.

Wanted: X_K from $e_{1:T}$ with $1 \leq K < T$.

Idea: „*Forward-Backward-Algorithm*“:

Filtering from 1 to K and „backward filtering“ from T to K.



[RN]

Procedure:

1. Split problem into forward- and backward-part.
2. Forward filtering: Already known.
3. Backward filtering: New.

Smoothing

Use evidences $e_{1:K}$ up to K and later ones $e_{K+1:T}$ separately:

$$\begin{aligned}
\mathbf{P}(X_K \mid e_{1:T}) &= \mathbf{P}(X_K \mid e_{1:K}, e_{K+1:T}) \\
&= \alpha\, \mathbf{P}(X_K \mid e_{1:K})\ \mathbf{P}(e_{K+1:T} \mid X_K, e_{1:K}) && \text{(Bayes)} \\
&= \alpha\, \mathbf{P}(X_K \mid e_{1:K})\ \mathbf{P}(e_{K+1:T} \mid X_K) && \text{(Markov sensor)} \\
&= \alpha\, f_{1:K}\, b_{K+1:T}
\end{aligned}$$

Backward recursion for $k = T-1, T-2, \ldots K+1, K$:

$$\begin{aligned}
\mathbf{P}(e_{k+1:T} \mid X_k) &= \Sigma_{x_{k+1}} \mathbf{P}(e_{k+1:T} \mid X_k, x_{k+1})\ \mathbf{P}(x_{k+1} \mid X_k) \\
&= \Sigma_{x_{k+1}} P(e_{k+1:T} \mid x_{k+1})\ \mathbf{P}(x_{k+1} \mid X_k) && \text{(cond. ind.)} \\
&= \Sigma_{x_{k+1}} P(e_{k+1}, e_{k+2:T} \mid x_{k+1})\ \mathbf{P}(x_{k+1} \mid X_k) \\
&= \Sigma_{x_{k+1}} P(e_{k+1} \mid x_{k+1})\ P(e_{k+2:T} \mid x_{k+1})\, \mathbf{P}(x_{k+1} \mid X_k)
\end{aligned}$$

Structure: $\boldsymbol{b_{k+1:T} = \text{Backward}(b_{k+2:T}, e_{k+1:T})}$ with $\boldsymbol{b_{k+1:T} = \mathbf{P}(e_{k+1:T} \mid X_k)}$

Smoothing

Rain – umbrella domain as before:

$\mathbf{P}(X_t \mid X_{t-1} = true) = \langle 0.7, 0.3 \rangle$ $\mathbf{P}(E_t \mid X_t = true) = \langle 0.9, 0.1 \rangle$

$\mathbf{P}(X_t \mid X_{t-1} = false) = \langle 0.3, 0.7 \rangle$ $\mathbf{P}(E_t \mid X_t = false) = \langle 0.2, 0.8 \rangle$

$\mathbf{P}(X_0) = \langle 0.5, 0.5 \rangle$; $e_1 = true$, $e_2 = true$. $\mathbf{P}(X_1 \mid e_{1:2}) = ?$

In general: $\mathbf{P}(X_K \mid e_{1:T}) = \alpha\, \mathbf{P}(X_K \mid e_{1:K})\, \mathbf{P}(e_{K+1:T} \mid X_K)$, here: $T=2$, $K=1$

Here: $\mathbf{P}(X_1 \mid e_{1:2}) = \alpha\, \mathbf{P}(X_1 \mid e_1)\ \ \mathbf{P}(e_2 \mid X_1)$

By filtering: $\mathbf{P}(X_1 \mid e_1) = \langle 0.818, 0.182 \rangle$

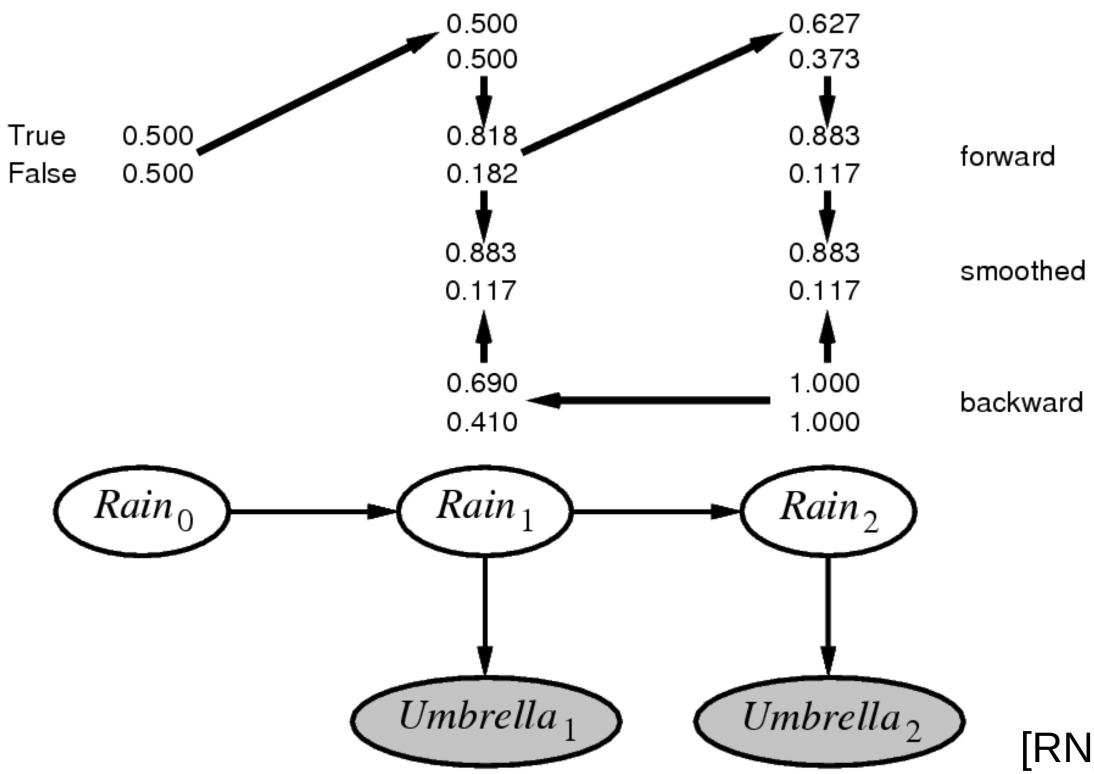
$$P(e_{k+1:T} \mid X_k) = \Sigma_{x_{k+1}} P(e_{k+1} \mid x_{k+1})\, P(e_{k+2:T} \mid x_{k+1})\ \mathbf{P}(x_{k+1} \mid X_k)$$

$$\mathbf{P}(e_2 \mid X_1) = \Sigma_{x_2}\ P(e_2 \mid x_2)\, P(e_{3:2} \mid x_2)\ \mathbf{P}(x_2 \mid X_1) \text{ with } P(e_{3:2} \mid x_2) = 1.$$

$$= 0.9 \cdot 1 \cdot \langle 0.7, 0.3 \rangle + 0.2 \cdot 1 \cdot \langle 0.3, 0.7 \rangle = \langle 0.69, 0.41 \rangle$$

$$\mathbf{P}(X_1 \mid e_{1:2}) = \alpha \langle 0.818, 0.182 \rangle \cdot \langle 0.69, 0.41 \rangle = \langle 0.883, 0.117 \rangle$$

Smoothing



[RN]

Most likely explanation

- Problem: Find the most likely explanation for a sequence of observed events.
- More precisely: Find the most likely sequence of hidden states that would cause the observed sequence of evidences.
- Example: For a boolean variable, for T steps there are 2^T possible sequences of states.
- Naive approach: Compute for each state in separation the probabilities using smoothing.
- But: The most likely sequence ≠ the sequence of most likely states!
- The most likely sequence requires maximizing the joint probability (not the isolated probabilities) !
- Solution: *Viterbi algorithm*.
- Applications: Cell phones, WLAN, hard disks, speech recognition.

Most likely explanation

Most likely path to x_{t+1} = most likely path to x_t plus another step:

$$\max_{x_1 \ldots x_t} \mathbf{P}(x_1, \ldots, x_t, X_{t+1} \mid e_{1:t+1}) =$$
$$\alpha\, \mathbf{P}(e_{t+1} \mid X_{t+1}) \quad \max_{x_t} [\, \mathbf{P}(X_{t+1} \mid x_t) \max_{x_1 \ldots x_{t-1}} P(x_1,\ldots,x_{t-1},x_t \mid e_{1:t})\,]$$

Like filtering $(f_{1:t+1} = \alpha\, \mathbf{P}(e_{t+1} \mid X_{t+1}) \ \Sigma_{x_t} \mathbf{P}(X_{t+1} \mid x_t) \ f_{1:t})$,

but:

1. $f_{1:t} = \mathbf{P}(X_t \mid e_{1:t})$ is replaced by

$$m_{1:t} = \max_{x_1 \ldots x_{t-1}} \mathbf{P}(x_1,\ldots,x_{t-1},X_t \mid e_{1:t}),$$

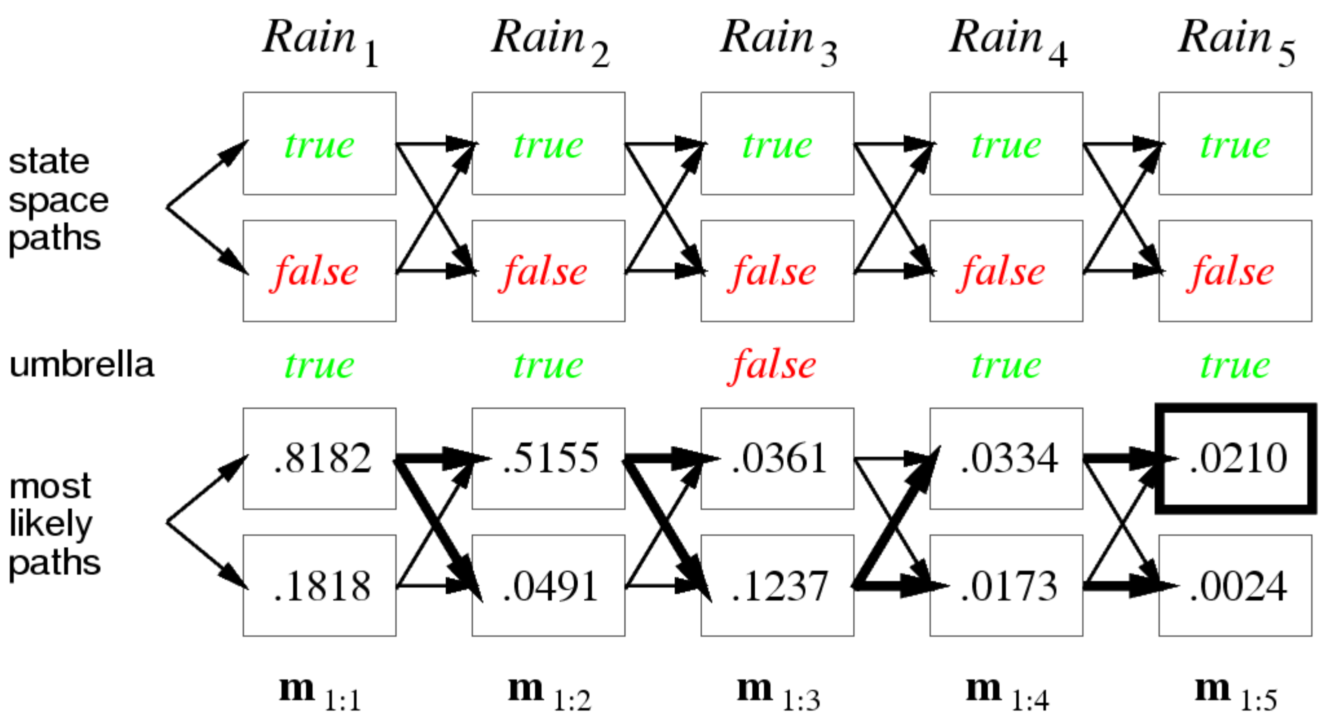
i.e. $m_{1:t}(i)$ is the probability of the most likely path to state i.

2. Replace sum over x_t by maximizing over x_t (Viterbi algorithm):

$$\mathbf{m_{1:t+1} = P(e_{t+1} \mid X_{t+1}) \ \max_{x_t} [\, P(X_{t+1} \mid x_t)\, m_{1:t}\,]}$$

Viterbi algorithm

1. Compute all $m_{1:t}$ successively. For each state, memorize the best previous state (thick arrows).
2. Choose the most likely state for time t.
3. Go back to the best previous state and so on.



[RN]

Hidden Markov Models

- So far: Transition model and sensor model were given by the experiment, no formal description.
- If a Markov process is described by states with a single variable:
  Hidden-Markov-Modell (*HMM*)
- Modeling temporal process and its evidences by two random processes:
  1. Random process: Markov chain with one hidden variable the transitions of which are desrcibed by probabilities.
  2. Random process: A Markov sensor provides evidences of the hidden variable.

Hidden Markov Models

Definition of a HMM:

- Let X_t be a single discrete random variable taking values (states) $\{s_1 \ldots s_n\}$, and
- E_t its evidence variable with values (possible observations) $\{e_1 \ldots e_m\}$. The matrix $T_{ij} = P(X_t = s_j \mid X_{t-1} = s_i)$ describes the probabilities for state transitions.
- The matrix $O_{ij} = P(e_j \mid s_i)$ is the observation matrix of probabilities that the Markov sensor yields observation e_j for state s_i.
- Starting distribution for X_0.

A HMM is stationary if T and O do not change over time.

With the

transition matrix $T_{ij} = \mathrm{P}(X_t = j \mid X_{t-1} = i)$ and the

observation matrix $(O_t)_{ii} = \mathrm{P}(e_t \mid X_t = i)$

we can simplify, e.g., smoothing using matrix notation:

$$f_{1:t+1} = \alpha\, O_{t+1} T^{\mathrm{T}} f_{1:t}$$

in place of

$$\mathbf{P}(X_{t+1} \mid e_{1:t+1}) = \alpha\, \mathbf{P}(e_{t+1} \mid X_{t+1}) \ \Sigma_{x_t} \mathbf{P}(X_{t+1} \mid x_t)\ \mathrm{P}(x_t \mid e_{1:t})$$

and

$$b_{k+1:t} = T O_{k+1}\, b_{k+2:t}$$

in place of

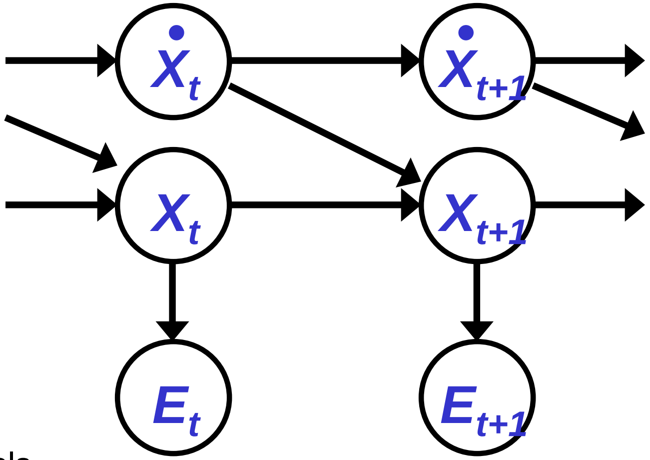
$$\mathbf{P}(e_{k+1:T} \mid X_k) = \Sigma_{x_{k+1}} \mathrm{P}(e_{k+1} \mid x_{k+1})\ \mathrm{P}(e_{k+2:T} \mid x_{k+1})\ \mathbf{P}(x_{k+1} \mid X_k).$$

Kalman filtering

- So far: No continuous variables.
- Kalman filtering provides a model for systems with continuous variables, in particular, time dependent variables.
- Example: Trajectory tracking. Position and its temporal derivative (velocity) are considered random variables.

  A bird is flying through a forest. Try to predict its trajectory though it is partially hidden behind trees.

- Other examples: Planets, robots, ecosystems, markets, fusion GPS – inertial sensors.
- Bayes network for linear dynamical system with position X_t and position measurement E_t :



Kalman filtering

Example: 1D trajectory

- Observe X-coordinate
- Observation at intervals Δt.
- Assumption: Velocity is approximately constant.

Simple trajectory prediction: $X_{t+\Delta t} = X_t + \dot{X}\,\Delta t$.

To account for measurement errors and non-constant velocity we assume an error with Gaussian distribution:

$$P(X_{t+\Delta t} = x_{t+\Delta t} \mid X_t = x_t, \dot{X}_t = \dot{x}_t) = N(x_t + \dot{x}\Delta t, \sigma, x_{t+\Delta t})$$

with $$N(x_0, \sigma, x) = \alpha \exp(-\tfrac{1}{2}\,(x - x_0)^2 / \sigma).$$

Kalman filtering: Adapting the Gaussians

Assumptions:

- Gaussian a-priori distribution
- Linear Gaussian transition model
- Linear Gaussian observation model.

Prediction:

If $\mathbf{P}(X_t \mid e_{1:t})$ has a Gaussian distribution, then the predicted distribution is also Gaussian:

$$\mathbf{P}(X_{t+1} \mid e_{1:t}) = \int_{x_t} \mathbf{P}(X_{t+1} \mid x_t)\, P(x_t \mid e_{1:t})\, dx_t$$

With $\mathbf{P}(X_{t+1} \mid e_{1:t})$ also we also have a Gaussian for

$$\mathbf{P}(X_{t+1} \mid e_{1:t+1}) = \alpha\, \mathbf{P}(e_{t+1} \mid X_{t+1})\, \mathbf{P}(X_{t+1} \mid e_{1:t}).$$

Hence, $\mathbf{P}(X_t \mid e_{1:t})$ is a multivariate Gaussian $N(\mu_t,\Sigma_t)$ for all t with mean μ and covariance matrix Σ.

Kalman filtering: Interpretation

- **P**$(X_t \mid e_{1:t})$ is (and stays!) Gaussian. Its parameters (mean, covariance) change over time.
- Thus **P**$(X_t \mid e_{1:t})$ can be described with the same number of parameters for all times t.
- As the Gaussian may become arbitrarily broad, the usable information on X may become very small, but …
- … at least, this small amount of usable information is still encoded in the same number of parameters.
- For the general case (non-linear, non-Gaussian) this does not hold: In general, the effort for the description of the posterior grows over time!

Kalman filtering: 1D random walk

- Gaussian random walk along X-axis, X_t is the random variable.
- Prior distribution (initial position measured with limited accuracy):

$$P(x_0) \quad = \quad N(\mu_0, \sigma_0, x_0).$$

- Transition model (walk along random path):

$$P(x_{t+1} \mid x_t) = N(x_t, \sigma_x, x_{t+1}).$$

- Observation model (position measurement with limited accuracy):

$$P(e_t \mid x_t) \quad = N(e_t, \sigma_e, x_t)$$

- First observation: e_1

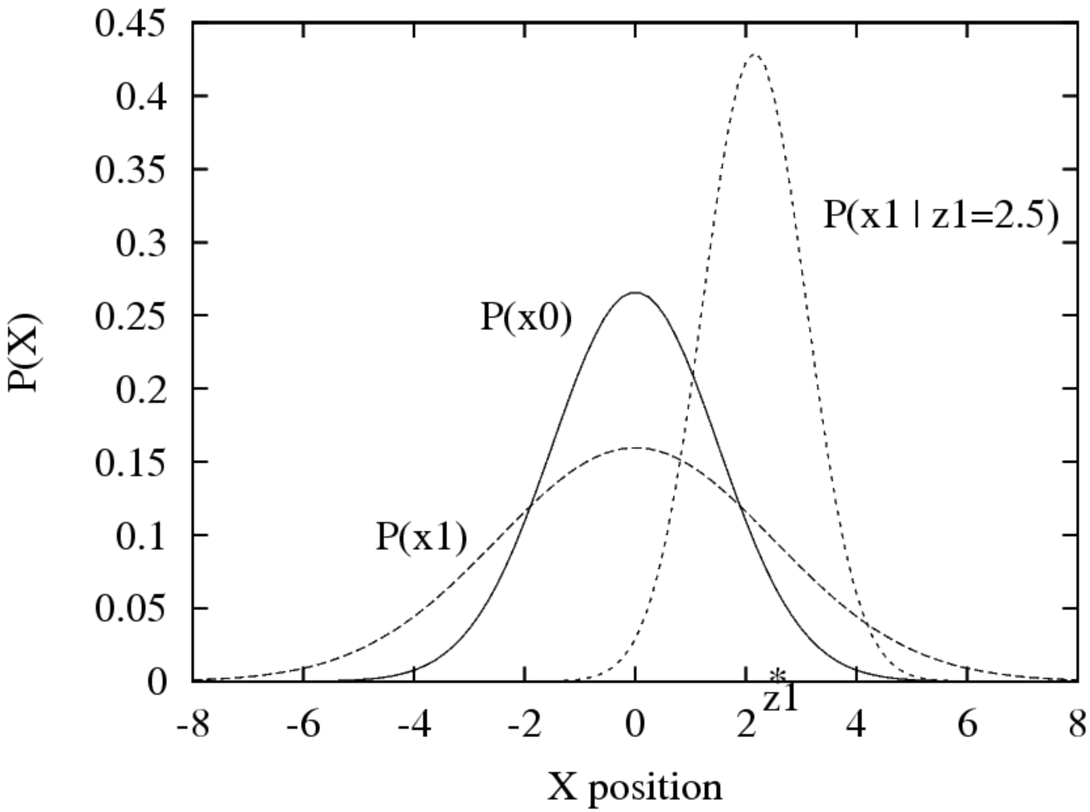
$$P(x_1 \mid e_1) \quad = \quad N(\mu_1, \sigma_1, x_1) \qquad \text{with} \qquad \mu_1 = \frac{(\sigma_0^2 + \sigma_x^2) e_1 + \sigma_e^2 \mu_0}{\sigma_0^2 + \sigma_x^2 + \sigma_e^2}, \quad \sigma_1^2 = \frac{(\sigma_0^2 + \sigma_x^2)\sigma_e^2}{\sigma_0^2 + \sigma_x^2 + \sigma_e^2}$$

- In general: $\mu_{t+1} = \dfrac{(\sigma_t^2 + \sigma_x^2) e_{t+1} + \sigma_e^2 \mu_t}{\sigma_t^2 + \sigma_x^2 + \sigma_e^2}, \qquad \sigma_{t+1}^2 = \dfrac{(\sigma_t^2 + \sigma_x^2)\sigma_e^2}{\sigma_t^2 + \sigma_x^2 + \sigma_e^2}$

Kalman filtering: 1D random walk

- Initial distribution: $\mu_0 = 0, \ \sigma_0 = 1$
- Transition cause by noise with $\sigma_x = 2$.
- Sensor noise: $\sigma_e = 1$.
- First observation: $e_1 = 2.5$.

- Prediction $P(x_1)$ is more flat than $P(x_0)$ due to the noisy transition.
- The mean μ_1 of $P(x_1 | e_1)$ is smaller than 2.5, because the prediction $P(x_1)$ is accounted for.



Kalman filtering: General case

Vector $\vec{x}$ of n random variables.

Vector of n observation values: $\vec{e}$.

Transition model: $\mathbf{P}(\vec{x}_{t+1} \mid \vec{x}_t) = N(F\vec{x}_t, \Sigma_x, \vec{x}_{t+1})$

Observation model: $\mathbf{P}(\vec{e}_t \mid \vec{x}_t) = N(H\vec{x}_t, \Sigma_e, \vec{e}_t)$

F: n x n - matrix of the linear transition model

H: n x n - matrix of the linear observation model

Σ_x: n x n - covariance matrix of the transition noise

Σ_e: n x n - covariance matrix of the observation noise

Gaussian with n variables:

$$N(\vec{\mu}, \Sigma, \vec{x}) = \alpha \exp\left(-\tfrac{1}{2} (\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})\right)$$

Kalman filtering: General case

Updating rule:

$\vec{\mu}_{t+1} = F\,\vec{\mu}_t + K_{t+1}\,(\vec{e}_{t+1} - H\,F\,\vec{\mu}_t)$

$\Sigma_{t+1} = (1 - K_{t+1})\,L$

with

$L = F\,\Sigma_t\,F^{\mathrm{T}} + \Sigma_x$

$K_{t+1} = L\,H^{\mathrm{T}}\,(H\,L\,H^{\mathrm{T}} + \Sigma_e)^{-1}$

K is the Kalman-Gain matrix.

Interpretation:

$F\,\vec{\mu}_t$: Predicted $\vec{\mu}$ (according to linear model)

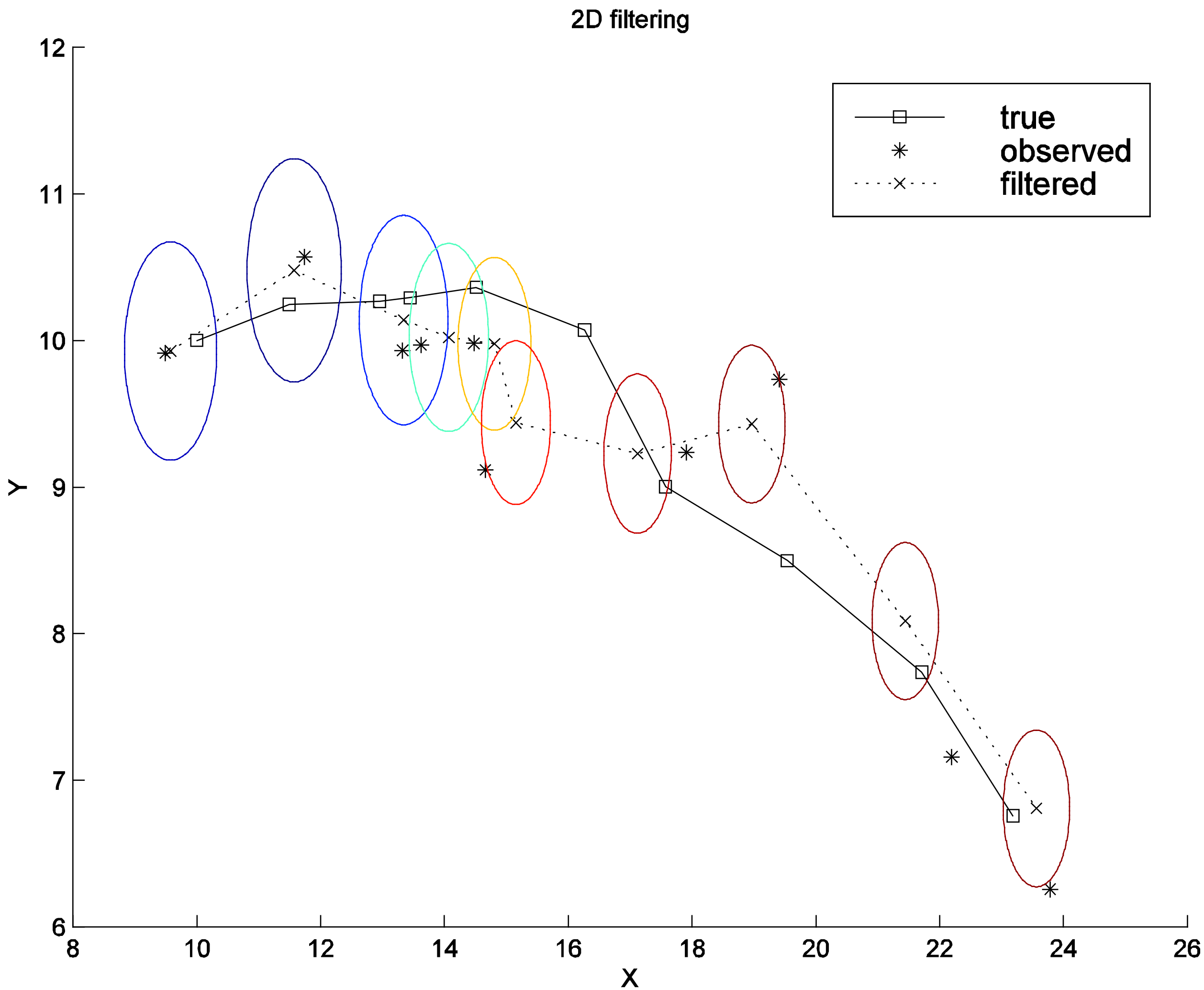
$H\,F\,\vec{\mu}_t$: Predicted observation

$e_{t+1} - HF\,\vec{\mu}_t$: Difference between prediction and observation

K_{t+1}: Confidence we have in the observation, used as a weight for comparison with the linear prediction

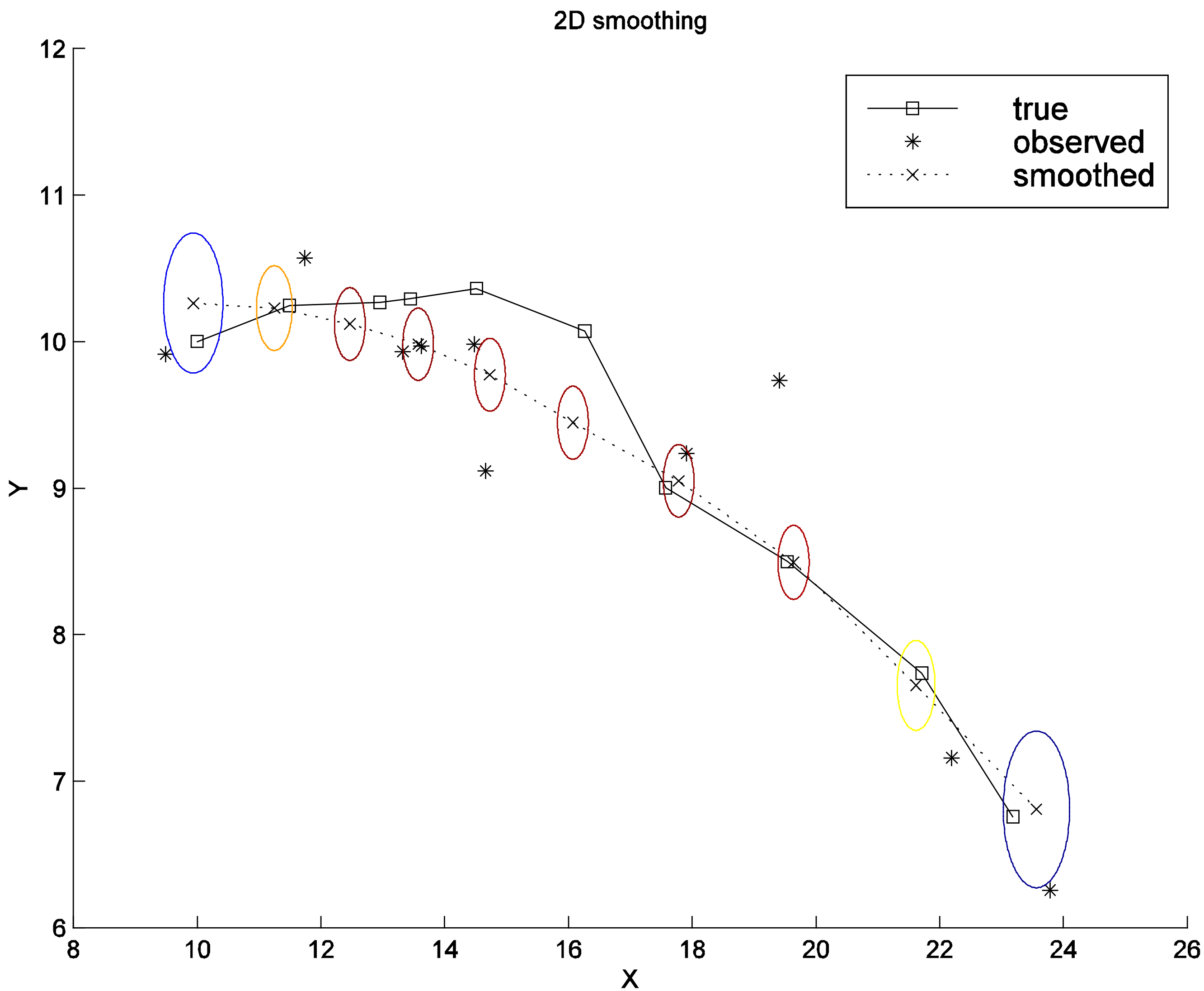
Σ_t and K_t are independent of the observed sequence and can thus be computed offline.

2D tracking: Filtering



$$\vec{X} = (X, \mathrm{d}X/\mathrm{d}t, Y, \mathrm{d}Y/\mathrm{d}t)^T$$

2D tracking: Smoothing



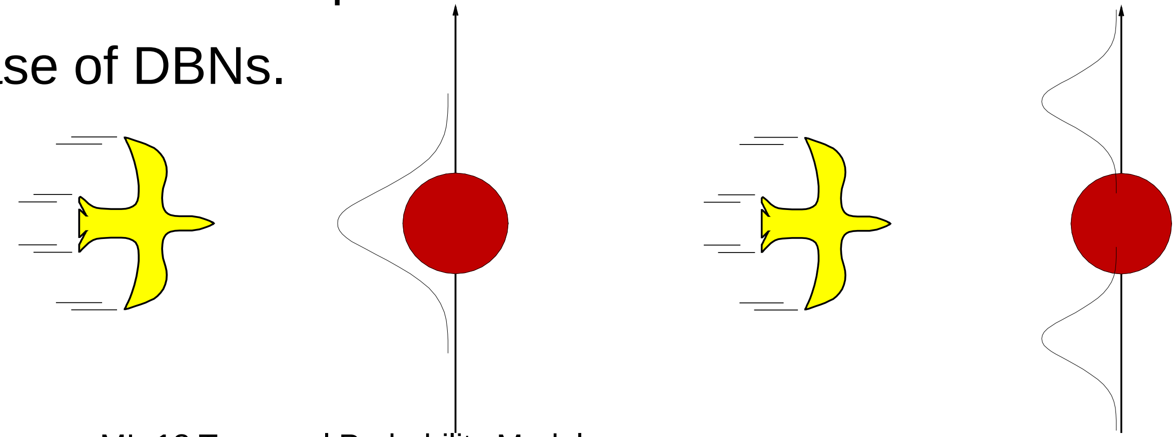
Kalman filtering: Limits

Simple Kalman filtering is not applicable if the transition model is non-linear.

Extension:

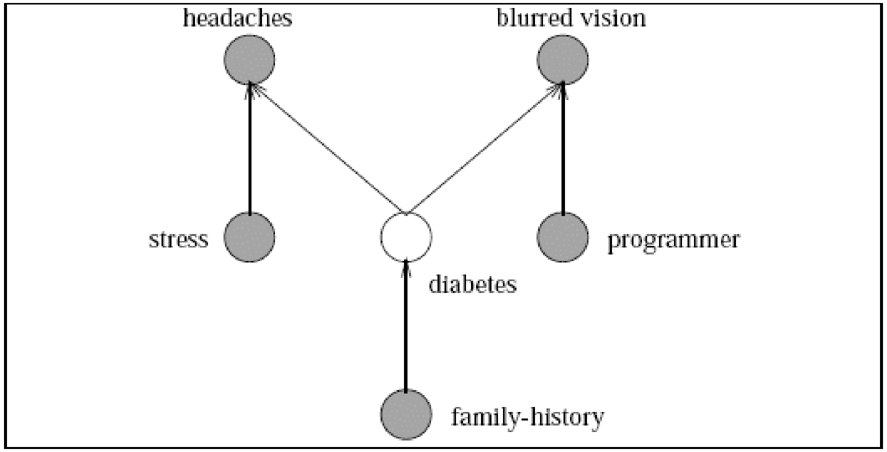
- Non-linearities can be treated by assuming local linearity in an environment of $x_t = \mu_t$.
- But this will fail if the system has a non-linearity at $x_t = \mu_t$.

  Example: Bird is flying towards a tree.
- Solution: Switching-Kalman-Filter
  - Applies several filters in parallel
  - Special case of DBNs.



Dynamic Bayesian networks

So far: Static Bayes networks for modeling dependencies without time:



[RN]

HMMs are a special case of of dynamic Bayesian networks (DBN) with just one variable.

Kalman filters are a special case with Gaussian distributions.

A general DBN is a temporal probability model with

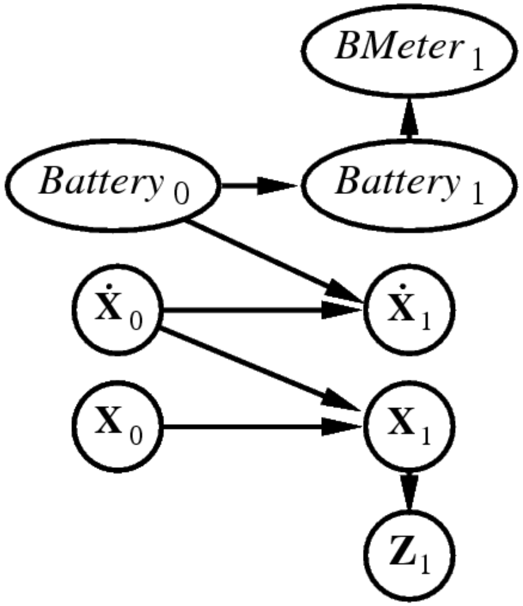
- an arbitrary number of random variables X_t, and
- evidence variables E_t
- for each time step.

Dynamic Bayesian networks

Example:

Robot with

- state variables position X_t, speed V_t, and battery power $Battery_t$;
- evidence variables measured position Z_t and $BMeter_t$.
- for t=0 and t=1.

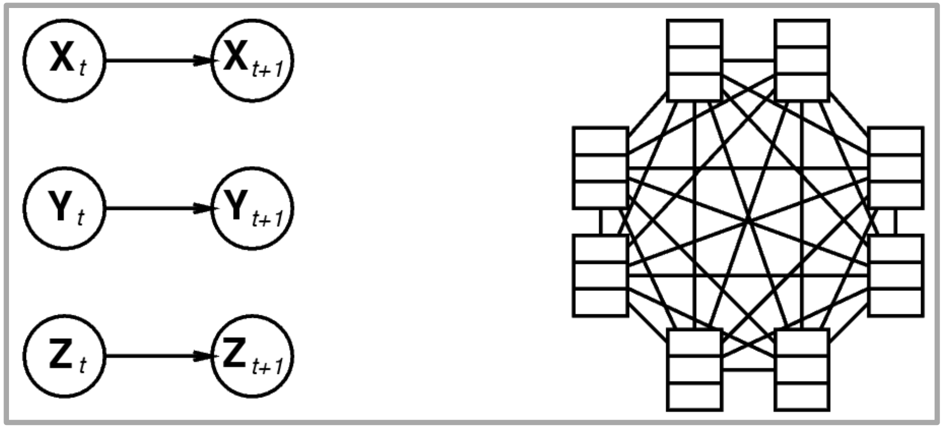


[RN]

DBNs and HMMs

- Every HMM is a DBN with just one variable.
- Every DBN with discrete variables can be represented as a HMM:
  - Combine all variables of the DBN to a single HMM-Variable.
  - The HMM-variable has one value for each combination of the variables of the DBN.
  - Problem: Combinatorial explosion.
- DBN are much better suited than HMMs as they employ „factorized“ states with an exponentially smaller number of parameters.

Example: 20 boolean state variables with 3 parents each. Parameters:
DBN 20 x 2^3 = 160,
HMM 2^{20} x 2^{20}.

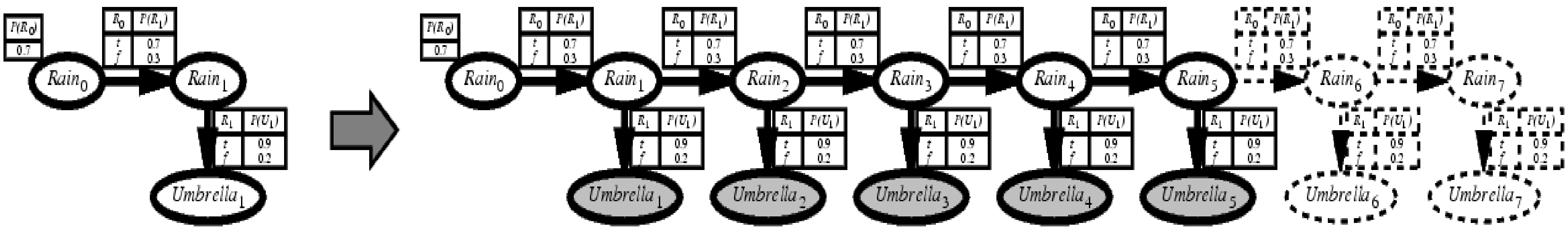


Inference in dynamic Bayesian networks

Naive method:

- Unroll DBN (represent each time step explicitly).
- Apply Algorithm for static Bayes nets.

Problem: Memory and computational effort $O(t)$.



[RN]

Alternative: Roll up filtering

Add step t+1, then sum out the variables of step t.

Only possible (for realistic size) with approximation methods such as particle filtering.

Summary

- Temporal probability models represent a domain using random variables for hidden states and observable evidences.
- Three assumptions
  - Markov process,
  - Markov sensor,
  - Stationary domain.
- Several types of inference: Filtering, prediction, smoothing, most likely explanation (Viterbi algorithm).
- HMMs model a Markov process using a single variable.
- Kalman filtering employs an arbitrary number of state variables but only Gaussian distribution.
- DBNs have an arbritrary number of variables and arbitrary distributions, but exact inference is infeasible due to computational effort. Particle filtering is a good approximation for filtering.

Speech Recognition

Speech recognition

- Speech recognition is an important application of temporal probability models.
- Recognize a sequence of words from a (raw) speech signal.
- Speech understanding:
  - Interpret sequence of words.
  - Find relation to other data, e.g., other sensors or a knowledge base.
- Speech signals are highly variable, ambiguous, noisy etc.
- Speech signals can not be classified after the simple scheme *signal* → *features* → *classifier* → *symbols*.
- Rather, simultaneous recognition on different levels of abstraction is required.

Speech recognition as probabilistic inference

Task:

What is the most likely word sequence given a signal?

→ Choose *Words* such that P(*Words* | *signal*) is maximized.

Bayes rule:

$$P(Words \mid signal) = \alpha\, P(signal \mid Words)\, P(Words).$$

Thus the problem is decomposed into an acoustic model and a language model.

Words are the hidden state sequence, *signal* is the observation (evidence) sequence.

Phones and Phonemes

- For classification, a small number of different entities and large number of training samples of each is required.
- English has about 700000 words,
- consisting of 10000 syllables,
- but these consist of only 40-50 phones (speech sounds).

Phones and Phonemes

- **Phones** are formed by the articulators (lips, teeth, tongue, vocal cords, air flow).
- Phones are closer to the signal than words.
  - ➔ Acoustic model = pronounciation model + phone model

- **Phonemes** are the smallest units that have an effect on meaning (they do not carry meaning in isolation).
- Phonemes are combined to the smalles meaningful units: Morphemes.
- Phonemes ≠ Characters
- Allophones are different speech sounds representing the same phoneme.
- Phonemes abstract phones to a representational level between signal and words.

Phones and Phonemes

DARPA-alphabet for American English (ARPAbet)

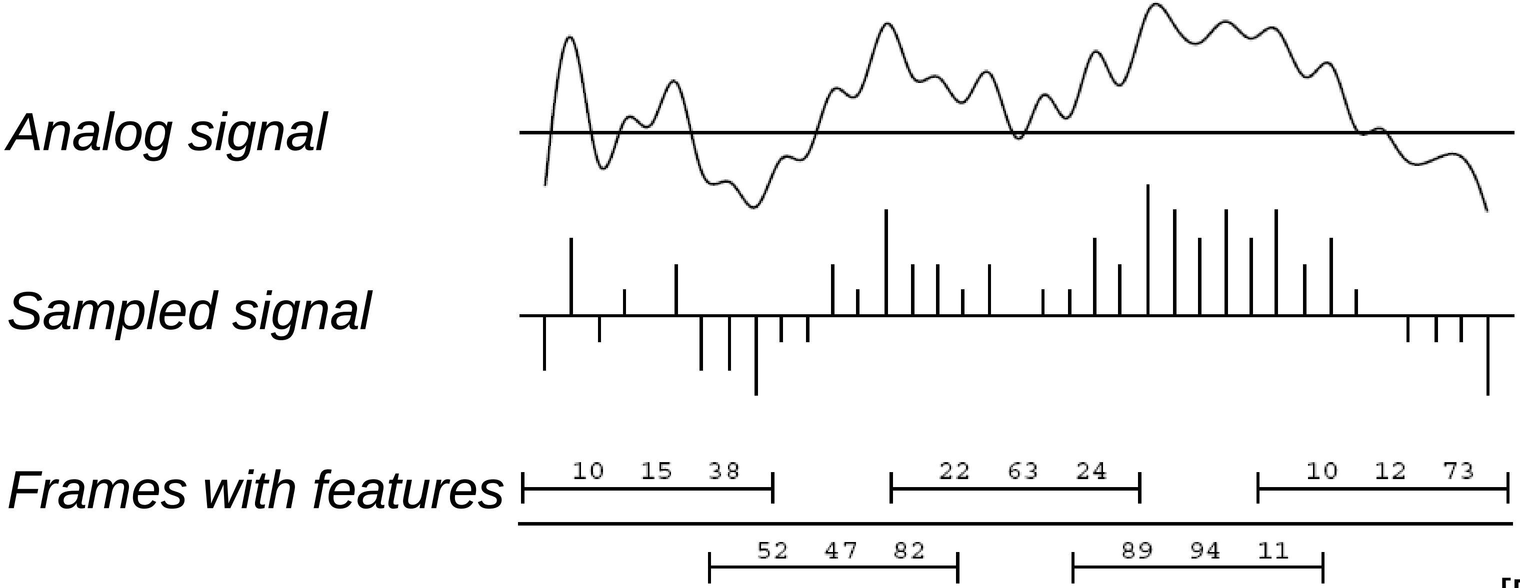
[iy]	beat	[b]	bet	[p]	pet
[ih]	bit	[ch]	Chet	[r]	rat
[ey]	bet	[d]	debt	[s]	set
[ao]	bought	[hh]	hat	[th]	thick
[ow]	boat	[hv]	high	[dh]	that
[er]	Bert	[l]	let	[w]	wet
[ix]	roses	[ng]	sing	[en]	button
⋮	⋮	⋮	⋮	⋮	⋮

[RN]

E.g. „ceiling“: [s iy l ich ng] / [s iy l ix ng] / [s iy l en]

Speech sounds

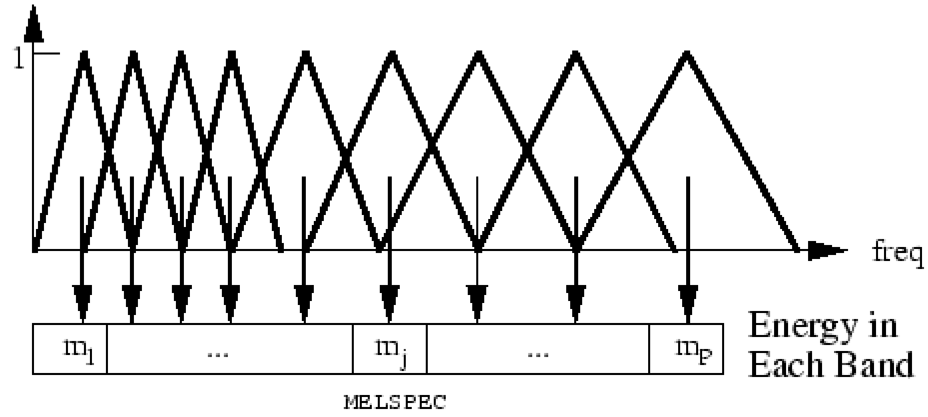
- Signal: Displacement of microphone membrane as a function of time.
- Representation: 8-16 kHz sampling, 8-12 bit quantization.
- Signal is processed in overlapping **frames** of 30 ms.
- Data reduction: Each frame is represented by features.
- Features: E.g., peaks of the power spectrum.



[RN]

Frames and features

- Overlap of frames 50%-75%.
- Features are, e.g., the distribution of energy over different frequencies, or change rates.
- Note energy distribution underlies uncertainty relation.
- Features may correlate with the activities of the articulators.

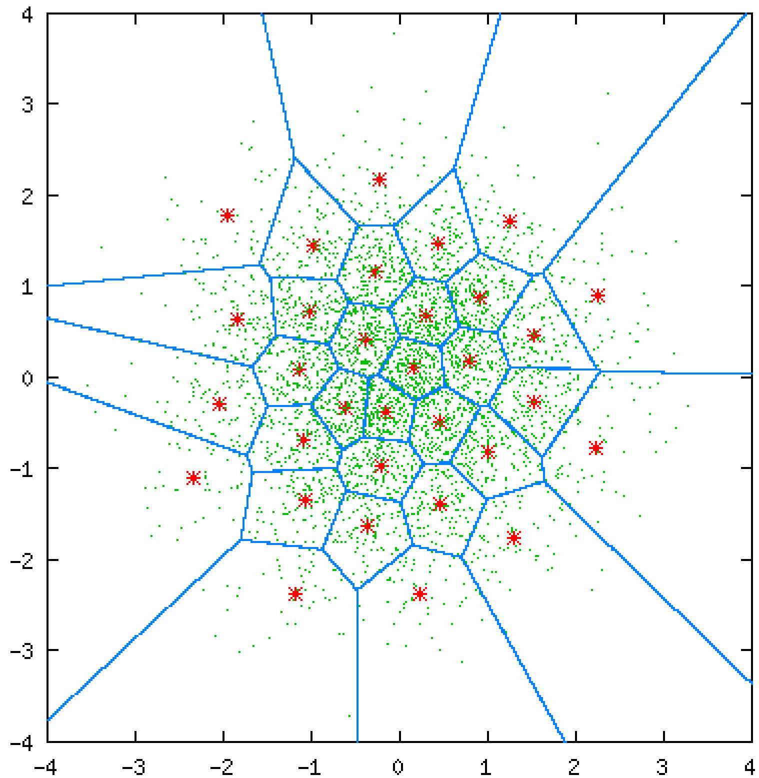


Phone models

- Frame features are vectors of high dimensionality, leaving still many options to encode a phone.
- *P(Features | Phone)* represents the frame features.
- Better and more compact representation by, e.g.,
  - natural numbers obtained from clustering, or
  - parameters of a Gaussian mixture model

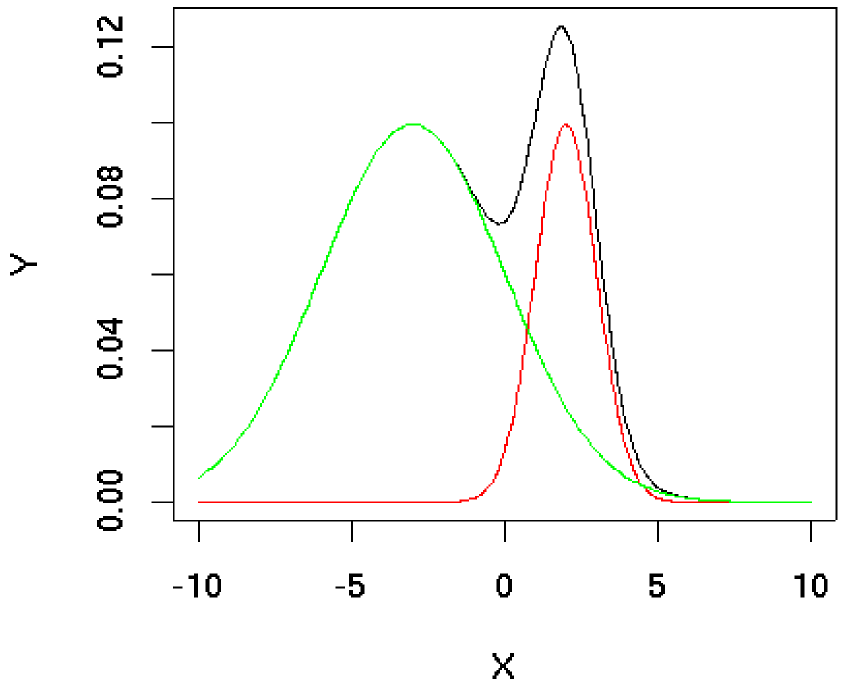
Phone models

Clustering can be used to form groups of frequent features, natural numbers denote the groups (centers):



Phone models

Gaussian mixtures describe *P*(*Features* | *Phone*) better than clusters.



¼ *N*(2,1) + ¾ *N*(-3,3)

Phone models

- Phones exhibit inner structure.
- This structure can be modeled effectively a three state phone model:
  - Each phone consists of *Onset, Mid, End*.
  - Example: [t] has silent *Onset*, explosive *Mid*, hissing *End*.
  - Thus *P*(*Features* | *Phone*) is replaced by

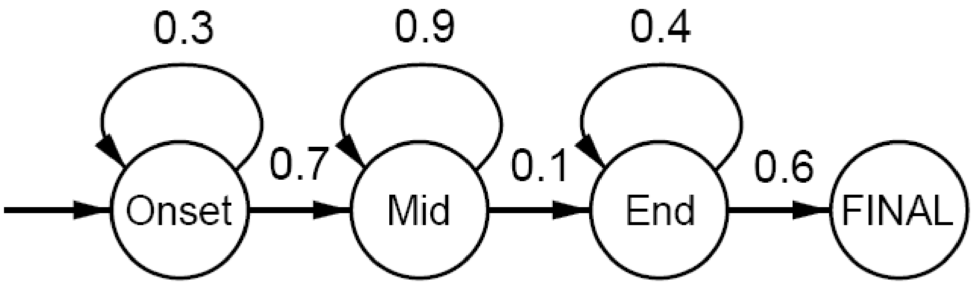
    P(*Features* | *Phone, Phase*).

Phone models

- Problem: Phones sound different, depending on neighboring speech sounds.
- These **coarticulation effects** come about because the articulators can not switch between positions instantaneously.
- Model for coarticulation: **Triphone context**
  - Each of n speech sounds is now represented by n^2 speech sounds which depend on both neighboring speech sounds.
  - Example: [t] in „star“ is represented by [t(s,aa)].
- Combining the three state model with the triphone model makes representation grow from n to n^3, but this is worth the expense.

Phone models

Phone HMM for [m]:



[RN]

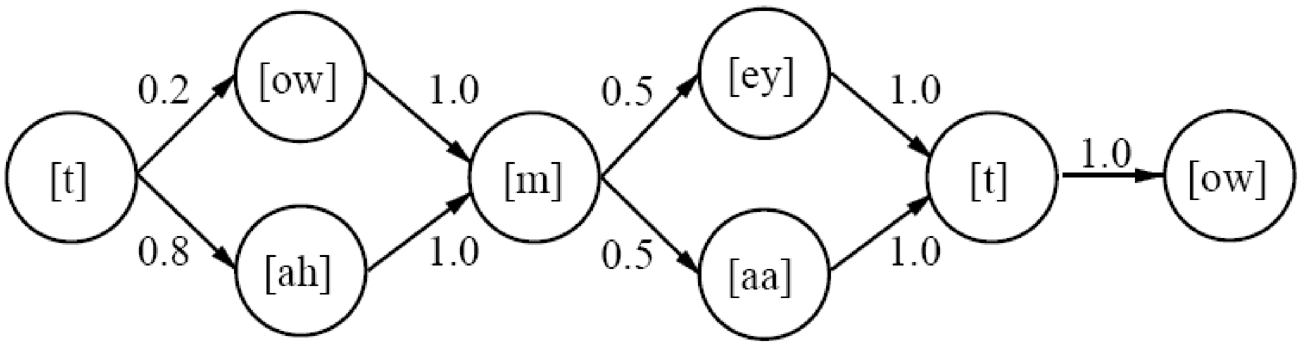
To each of the three states of the Phone HMM belong output probabilities for the features (e.g., cluster numbers):

Onset:	Mid:	End:
C1: 0.5	C3: 0.2	C4: 0.1
C2: 0.2	C4: 0.7	C6: 0.5
C3: 0.3	C5: 0.1	C7: 0.4

[RN]

Word pronounciation models

A word is represented by a probability distribution over a phone sequence. This sequence is represented by a HMM:



[RN]

P([towmeytow] | „tomato“) = P([towmaatow] | „tomato“) = 0.1

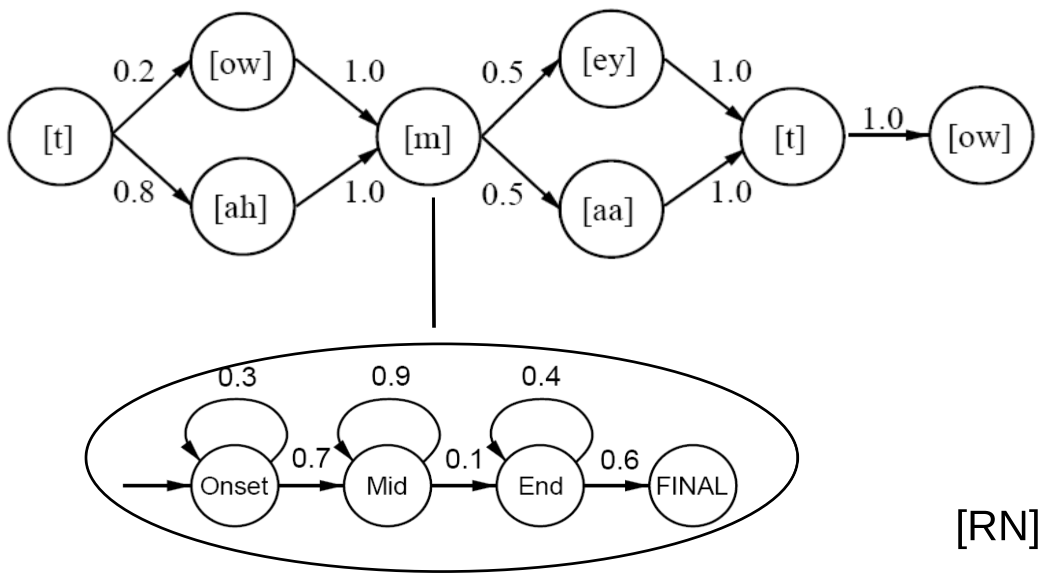
P([tahmeytow] | „tomato“) = P([tahmaatow] | „tomato“) = 0.4

- The structure of the HMM is created manually.
- Transition probabilities are estimated from data.

Word models

A **word model** consists of the phone models and the pronounciation model.

Word model for *Tomato*:



[RN]

State of a word HMM = phone + phone state,

e.g., the word HMM of *Tomato* has a state $[m]_{Mid}$.

Recognition of isolated words

Phone models + word models fix $P(e_{1:t} \mid Word)$ for isolated words.
where $e_{1:t}$ are the observed features.

Recognizing a word means maximizing

$$P(Word \mid e_{1:t}) = \alpha\, P(e_{1:t} \mid Word)\, P(Word),$$

where the prior $P(Word)$ is just obtained from the word frequencies.

$P(e_{1:t} \mid Word)$ is computed recursively by

$$\mathbf{P}(X_{t+1}, e_{1:t+1}) = \text{Forward}(\mathbf{P}(X_t, e_{1:t}), e_{t+1})$$

and $$P(e_{1:t} \mid Word) = \sum_{x_t} P(x_t, e_{1:t}).$$

Recognition of isolated words (e.g. for dictation) reaches 95-99% accuracy (with training on a particular person).

Recognition of continuous speech

Recognition of continuous speech ≠ recognition of sequence of isolated words, because

- adjacent words are strongly correlated,
- the most likely sequence of words ≠ the sequence of most likely words,
- segmentation of words is difficult, because there are few gaps between words which become visible on the signal level (only on the high level of human speech processing),
- there is cross-word coarticulation, e.g., „next thing“.

Recognition of continuous speech manage 60-80% accuracy.

Language model

A language model specifies the a priori probability of each sequence of words using the chain rule:

$$P(w_1 \ldots w_n) = \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1}).$$

Most factors are hard to estimate.

Bigram model as an approximation:

$$P(w_i \mid w_1 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1}),$$

i.e., first order Markov assumption.

Training: Count all word pairs in a large text corpus.

More complex models such as Trigrams

$$P(w_i \mid w_1 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1}, w_{i-2}),$$

or grammars lead to some improvement.

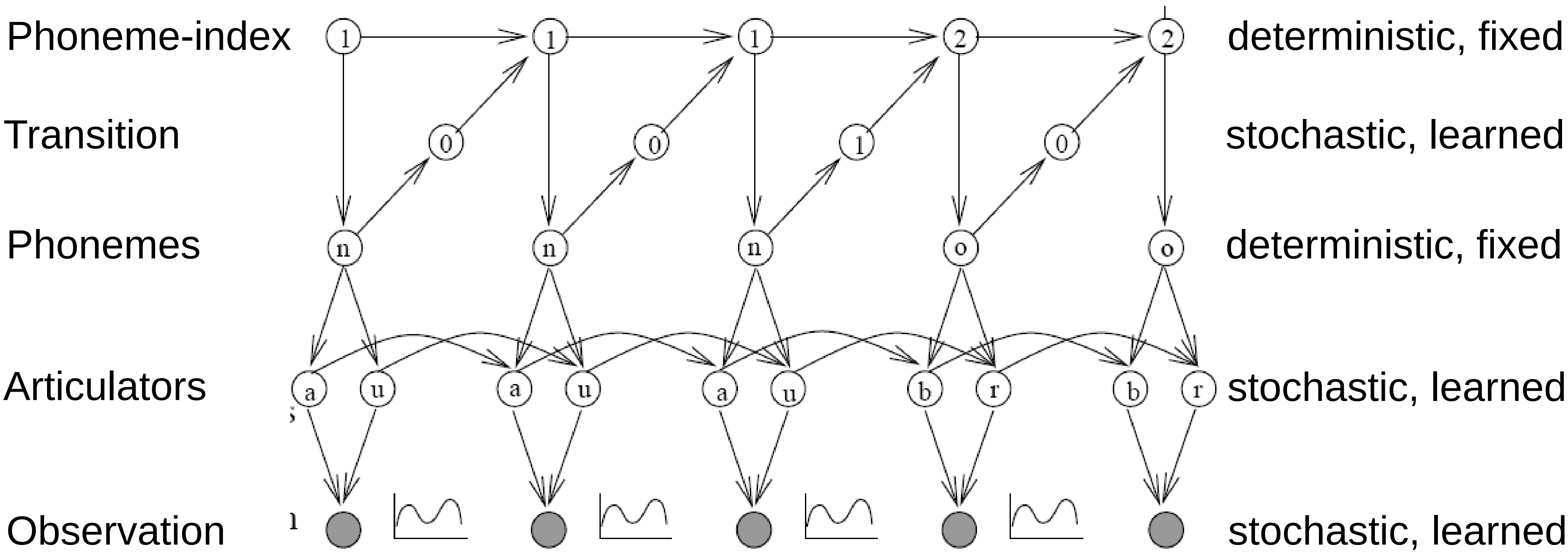
Recognition of word sequences: Combined HMM

- Combine **word model** and Bigram **language model** to an HMM.
- States of the combined HMM are specified by word, phone and phone state.
- Example: $[m]^{Tomato}_{Mid}$.
- Transitions:
  - Phone state – phone state (within a phone),
  - Phone – phone (within a word),
  - Word final state – word initial state (between words).
- Representational effort:
  The combined HMM for W words with an average of L three state phones has $3LW$ states.

Recognition of word sequences: Combined HMM

- Most likely phone sequence is found by Viterbi algorithm
  - this fixes also the word sequence and
  - solves the segmentation problem.
- But: The word sequence obtained from the most likely phone sequence is not necessarily the most likely word sequence …
- … because probability of a word sequence = sum of probabilities of all corresponding state sequences.
- Solution: **A*-decoder** to find the most likely word sequence with moderate computational effort (Jelinek 1969).

DBNs for speech recognition



- Further variables for gender, accent, speed are easy to add.
- Better performance than HMMs.

Summary

- Speech recognition has been formulated as probabilistic inference since 70ies.
- Evidence = speech signal
- Hidden variables = phone and word sequences
- Context effects such as coarticulation are handled by augmenting the states.
- Highly successful approach.

Image sources

[M] Online material available at www.cs.cmu.edu/~tom/mlbook.html for the textbook: Tom M. Mitchell: *Machine Learning*, McGraw-Hill

[RN] Stuart Russell, Peter Norvig: *Artificial Intelligence*, Pearson

[H] Gunther Heidemann, 2012.

- p-norm unit circles
- Optimization based clustering
- K-Means
- Conceptual clustering
- Hebbian Learning:
  - Hebb rule
  - Anti-Hebb rule
- Eigenfaces
- Principal curves and SOM

- MLP
  - Parameters
  - Comparison to RBF
- RBF
  - Parameters
- SOM
- Q-Learning: Probabilistic choice of actions