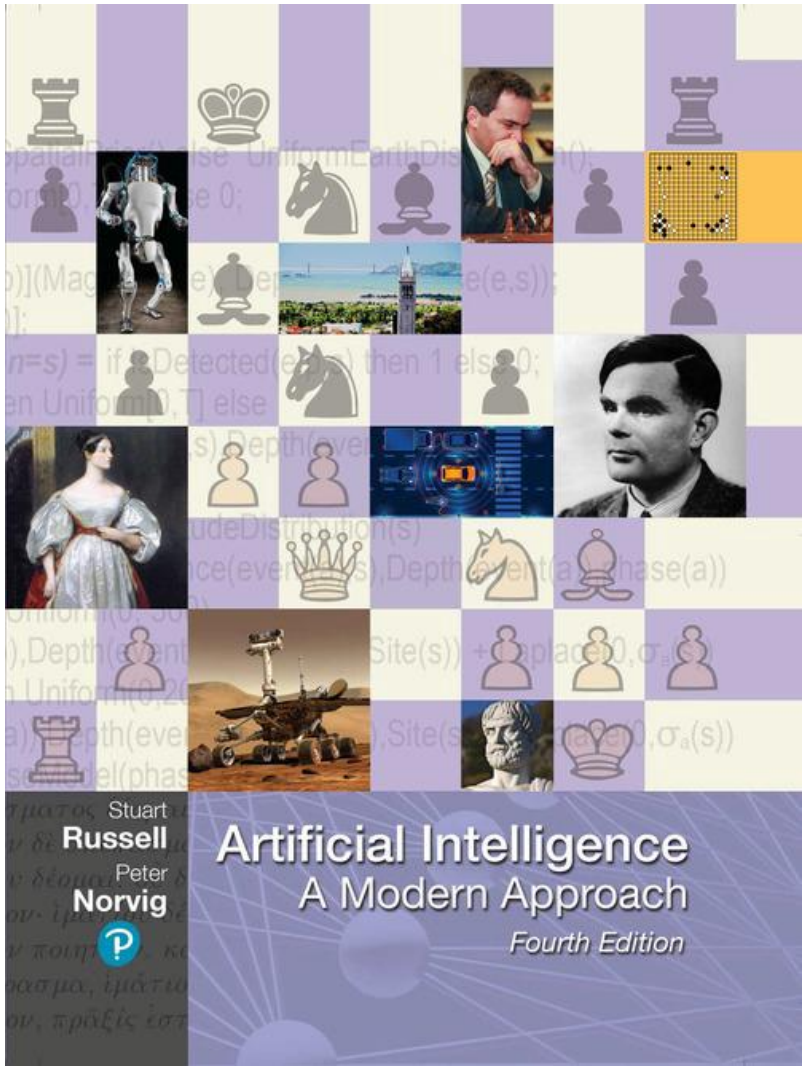
Artificial Intelligence Fundamentals

2024-2025



"Probability theory is nothing but common sense reduced to calculation."

- Pierre-Simon Laplace

AIMA Chapter 14

Probabilistic Reasoning Over Time

Outline

- Time and Uncertainty
- Inference in Temporal Models
- Hidden Markov Models
- Predicting Market Volatility with HMMs

Motivation

We have developed our techniques for probabilistic reasoning in the context of **static worlds**, in which each random variable has a **single fixed value**.

Example: "*repairing a car*":

- we assume that whatever is broken remains broken during the process of diagnosis;
- our job is to infer the state of the car from observed evidence, which also remains fixed.

Example: "*treating a diabetic patient*":

- The task is to assess the current state of the patient, including the actual blood sugar level and insulin level. Given this information, we can *make a decision* about the patient's food intake and insulin dose.

- dynamic aspects of the problem are essential. Blood sugar levels and measurements thereof **can change rapidly over time**, depending on recent food intake and insulin doses, metabolic activity, the time of day, and so on.

- To assess the current state from the **history of evidence** and to predict the outcomes of treatment actions, we must model these changes.

Time and Uncertainty

States and observations

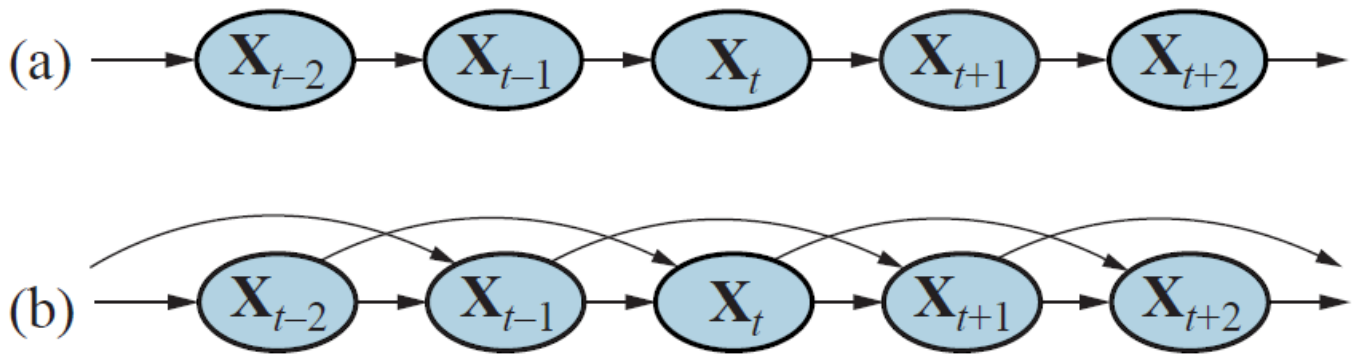
- discrete-time models, in which the world is viewed as a series of **snapshots** or **time slices**
- the time interval Δ between slices is assumed to be **the same** for every interval.
- $\mathbf{X}_t$: denotes the set of state variables at time t, which are assumed to be unobservable
- $\mathbf{E}_t$: denotes the set of observable evidence variables.
- The observation at time t is $\mathbf{E}_t = \mathbf{e}_t$

Transition and sensor models

- The **transition model** specifies the probability distribution over the latest state variables, given the previous values: $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{0:t-1})$.
- Problem: the set $\mathbf{X}_{0:t-1}$ is unbounded in size as t increases.
- Solution: **Markov assumption** [the current state depends on only a **finite fixed** number of previous states]
- **First order**, $\boldsymbol{P}(\boldsymbol{X}_t \mid \mathbf{X}_{t-1})$; **Second order**, $\mathbf{P}(\mathbf{X}_t \mid \mathbf{X}_{t-2}, \mathbf{X}_{t-1})$; …
- $\boldsymbol{P}(\boldsymbol{E}_t \mid \mathbf{X}_t)$ is our **sensor model**, sensor Markov assumption:

$$\boldsymbol{P}(\boldsymbol{E}_t \mid \mathbf{X}_{0:t}, \mathbf{E}_{1:t-1}) = \mathbf{P}(\mathbf{E}_t \mid \mathbf{X}_t)$$

Time and Uncertainty



(a) Bayesian network structure corresponding to a first-order Markov process with state defined by the variables $\mathbf{X}_t$

(b) A second-order Markov process.

Time and Uncertainty

- the prior probability distribution at time 0, $\mathbf{P}(\mathbf{X}_0)$.

$$\mathbf{P}(\mathbf{X}_{0:t}, \mathbf{E}_{1:t}) = \mathbf{P}(\mathbf{X}_0) \prod_{i=1}^{t} \mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1}) \mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i) .$$

- The first-order Markov assumption says that the state variables contain all the information needed to characterize the probability distribution for the next time slice.
- Ways to improve the accuracy of the approximation
  - **Increasing the order** of the Markov process mode
  - **Increasing the set of state** variables

Umbrella World

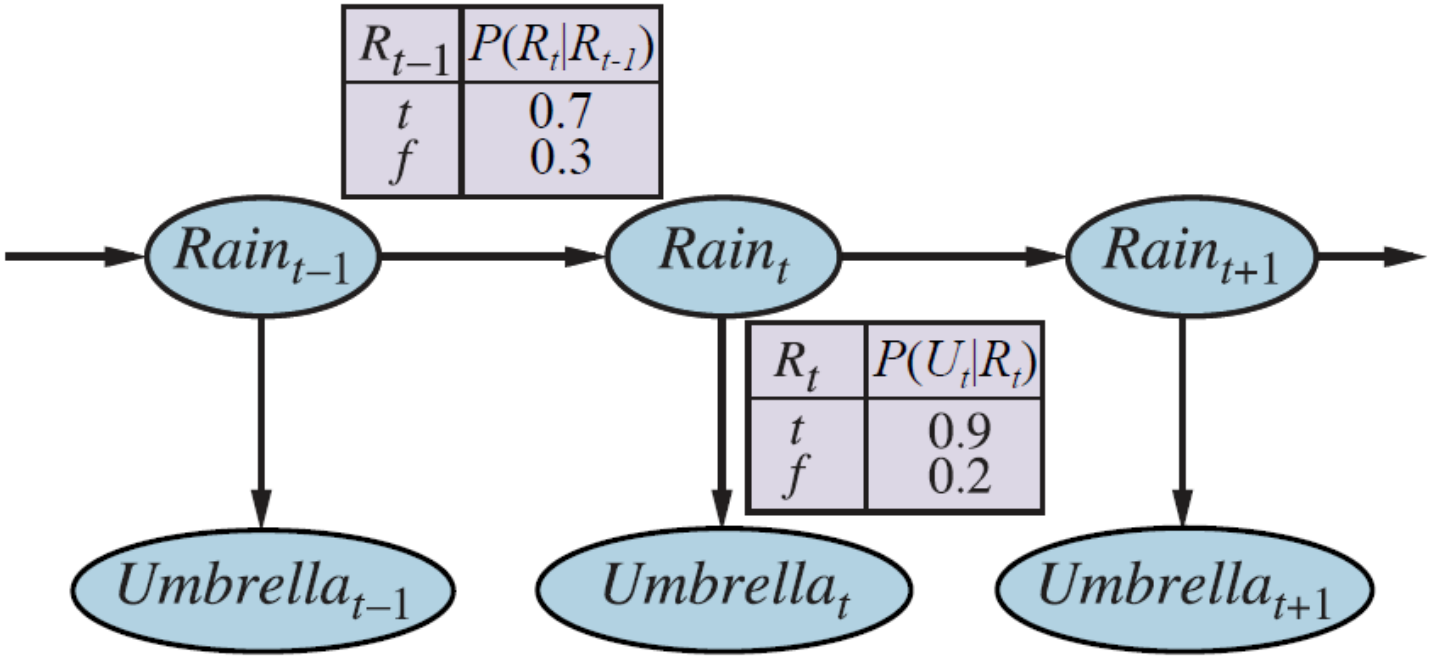
You are the security guard stationed at a secret underground installation.

You want to know whether it's raining today, but your only access to the outside world occurs each morning when you see the **director** coming in with, or without, an umbrella.

For each day t, the set $\mathbf{E}_t$ thus contains a single evidence variable $\mathbf{Umbrella}_t$ or $\mathbf{U}_t$ for short (*whether the umbrella appears*), and the set $\mathbf{X}_t$ contains a single state variable $\mathbf{Rain}_t$ or $\mathbf{R}_t$ for short (*whether it is raining*)

We will assume that the state sequence starts at t=0 and evidence starts arriving at t =1.

Umbrella World



Bayesian network structure and conditional distributions describing the umbrella world. The transition model is $\mathbf{P}(Rain_t \mid Rain_{t-1})$ and the sensor model is $\mathbf{P}(Umbrella_t \mid Rain_t)$.

Stationary vs Non-stationary Environment

Hence, we assume a **stationary process**, i.e. the conditional probability distribution is the same for all t.

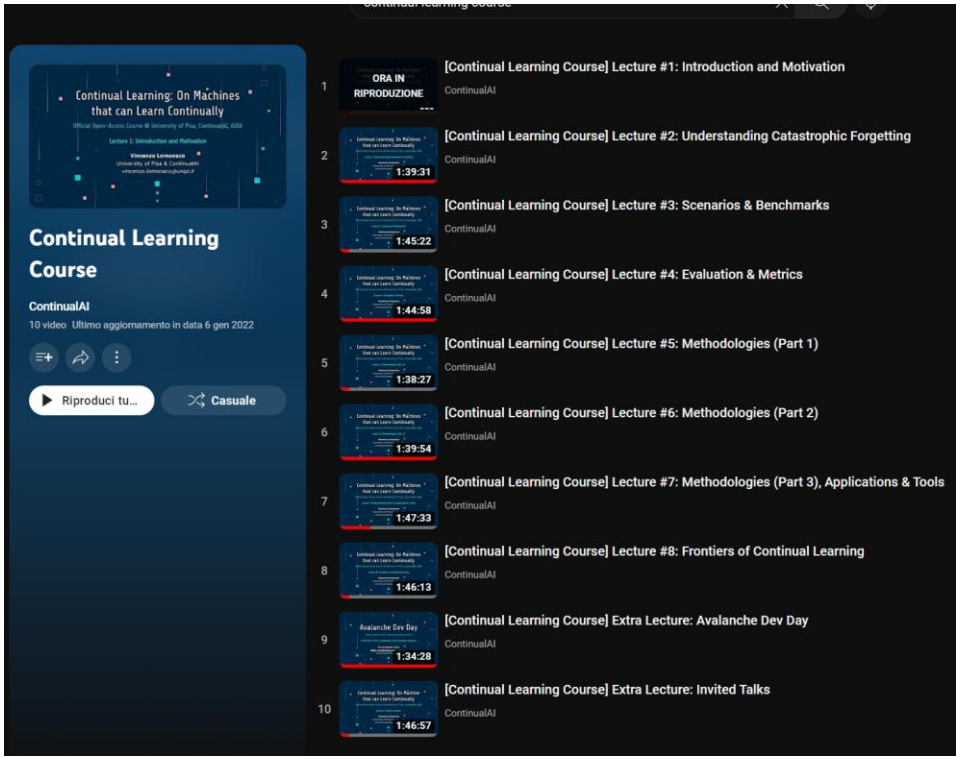
$\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ is the same for all t $\mathbf{P}(R_t|r_{t-1}) = <0.7, 0.3>$

Change is governed by laws that **do not themselves change** over time.

Learning in non-stationary environments ->

Focus of *Continual Learning!*

[Continual Learning Course - YouTube]



Inference in Temporal Models

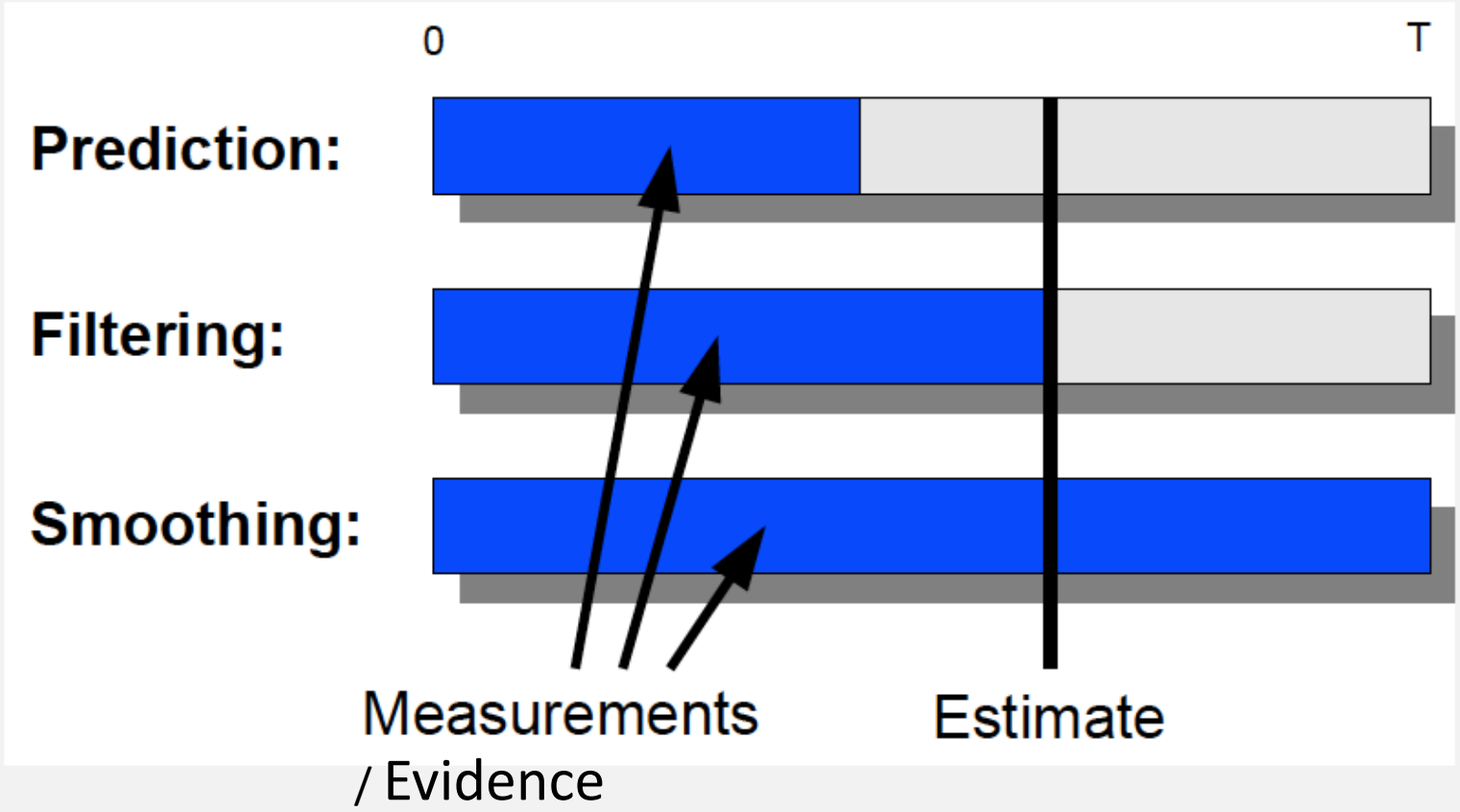
Basic inference tasks that must be solved:

- **Filtering** (or **state estimation**) is the task of computing the **belief state** $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$
- **Prediction**: This is the task of computing the posterior distribution over a future state, given all evidence to date.
- **Smoothing**: This is the task of computing the posterior distribution over a past state, given all evidence up to the present
- **Most likely explanation**: Given a sequence of observations, we might wish to find the sequence of states that is most likely to have generated those observations

Besides inference tasks:

- **Learning:** The transition and sensor models, if not yet known, can be learned from observations

Inference in Temporal Models



From a tutorial by Simo Särkkä

Filtering

A good filtering algorithm maintains a **current state estimate** and updates it, rather than going back over the entire history of percepts for each time

The filtering function f takes into account **the state estimation computed up to the present** and the **new evidence.**

$$\mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = f(\boldsymbol{e}_{t+1}, \mathbf{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t}))$$

This process is called **recursive estimation** and is made of two parts:

1. **Prediction**: the current state distribution is projected forward from t to $t+1$: $\mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$
2. **Update**: then it is updated using the new evidence e_{t+1}: $\mathbf{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1})$

Reminder

$$\mathbf{P}(X \mid Y, Z) = \frac{\mathbf{P}(Y \mid X, Z)\ \mathbf{P}(Y \mid Z)}{\mathbf{P}(X \mid Z)}$$ (*Bayes rule with background knowledge Z*)

In our case:

$$\mathbf{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1}) = \alpha\ \mathbf{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1}\ \boldsymbol{e}_{1:t})\ \mathbf{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t})$$

Conditioning is marginalization applied to conditional probabilities

$$\mathbf{P}(\mathbf{Y}) = \sum_{Z \in \mathbf{Z}} \mathbf{P}(\mathbf{Y}, \boldsymbol{z}) \quad \text{--- product rule ---} \quad = \sum_{Z \in \mathbf{Z}} \mathbf{P}(\mathbf{Y} \mid \boldsymbol{z})\ P(\mathrm{z}) \qquad (\textit{Conditioning})$$

In our case:

$$\mathbf{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{e}_{1:t}) = \sum_{\boldsymbol{x}_t} \mathbf{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t, \boldsymbol{e}_{1:t})\ \mathbf{P}(\boldsymbol{x}_t \mid \boldsymbol{e}_{1:t})$$

Filtering

It can be computed as follows:

$$\mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t}, \boldsymbol{e}_{t+1}) =$$ *Dividing up the evidence*

$$= \alpha\, \mathbf{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}, \boldsymbol{e}_{1:t})\, \mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$$ *Bayes's rule, focusing on* $\boldsymbol{e}_{t+1}$ →

$$= \alpha\, \mathbf{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1})\, \mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t})$$ *Markov's sensor assumption*

$$= \alpha\, \mathbf{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \sum_{x_t} \mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t, \boldsymbol{e}_{1:t})\, \mathbf{P}(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})$$ *By conditioning on the current state* $\boldsymbol{X}_t$ →

$$= \alpha\, \mathbf{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \sum_{x_t} \mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)\, \mathbf{P}(\boldsymbol{x}_t|\boldsymbol{e}_{1:t})$$ *Markov's sensor assumption*

Finally:

$$\mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{e}_{1:t+1}) = \alpha\, \mathbf{P}(\boldsymbol{e}_{t+1}|\boldsymbol{X}_{t+1}) \sum_{x_t} \mathbf{P}(\boldsymbol{X}_{t+1}|\boldsymbol{x}_t)\, \mathbf{P}(\boldsymbol{x}_t|\boldsymbol{e}_{1:t}) \quad (\textit{Filtering})$$

----update--- *---one-step prediction----*

We can think of $\mathbf{P}(\boldsymbol{X}_t|\boldsymbol{e}_{1:t})$ as a *message* $f_{1:t}$ that is propagated forward in the sequence.
This makes evident the recursive structure:

$f_0 = \mathbf{P}(\boldsymbol{X}_0)$
$f_{1:t+1} = \alpha\, \textit{Forward}(f_{1:t}, \boldsymbol{e}_{t+1})$

where *Forward* implements the *filtering* update in **costant time**

Filtering in the Umbrella World

The example shows the consequences of observing umbrellas for two consecutive days, $\mathbf{P}(R_2|u_1,u_2)$

f_0

Day 0: $\mathbf{P}(R_0) = \langle 0.5, 0.5 \rangle$ no observation, a 50% chance of rain

Day 1:

Predict: $\mathbf{P}(R_1) = \sum_{r_0} \mathbf{P}(R_1|r_0)\, P(r_0) = \mathbf{P}(R_1|r_0) \times P(r_0) + \mathbf{P}(R_1|\neg r_0) \times P(\neg r_0) =$
$= \langle 0.7, 0.3 \rangle \times 0.5 + \langle 0.3, 0.7 \rangle \times 0.5 = \langle 0.5, 0.5 \rangle$

Update, observing the evidence u_1 ($U_1 = \text{true}$)

$f_{1:1}$

$\mathbf{P}(R_1|u_1) = \alpha\, \mathbf{P}(u_1|R_1)\, \mathbf{P}(R_1) = \alpha \langle 0.9, 0.2 \rangle \langle 0.5, 0.5 \rangle = \alpha \langle 0.45, 0.1 \rangle \approx \langle 0.818, 0.182 \rangle$

Day 2:

Predict: $\mathbf{P}(R_2) = \sum_{r_1} \mathbf{P}(R_2|r_1)\, P(r_1) = \langle 0.7, 0.3 \rangle \times 0.818 + \langle 0.3, 0.7 \rangle \times 0.182 = \langle 0.627, 0.373 \rangle$

Update, observing the evidence u_2

$f_{1:2}$

$\mathbf{P}(R_2|u_1,u_2) = \alpha\, \mathbf{P}(u_2|R_2)\, \mathbf{P}(R_2|u_1) = \alpha \langle 0.9, 0.2 \rangle \langle 0.627, 0.373 \rangle = \alpha \langle 0.565, 0.075 \rangle \approx \langle 0.883, 0.117 \rangle$

The **probability of rain increases** from day 1 to day 2 because rain persists.

Prediction

The task of prediction can be seen simply as filtering without the contribution of new evidence. The filtering process already incorporates a **one-step prediction**.

In general, looking ahead k steps, at time $t+k+1$, given evidence up to t:

$$\mathbf{P}(\boldsymbol{X}_{t+k+1}|\boldsymbol{e}_{1:t}) = \sum_{X_{t+k}} \mathbf{P}(\boldsymbol{X}_{t+k+1}|x_{t+k})\ \mathbf{P}(x_{t+k}|\boldsymbol{e}_{1:t}) \qquad \textit{... looking k steps ahead}$$

This computation involves only *the transition model* and **not** the *sensor model*.

We can show that the predicted distribution for *Rain* converges to a fixed point $\langle 0.5, 0.5 \rangle$, after which it remains constant for all time (the **stationary distribution** of the Markov process).

The ***mixing time*** is the time to reach the fixed point.

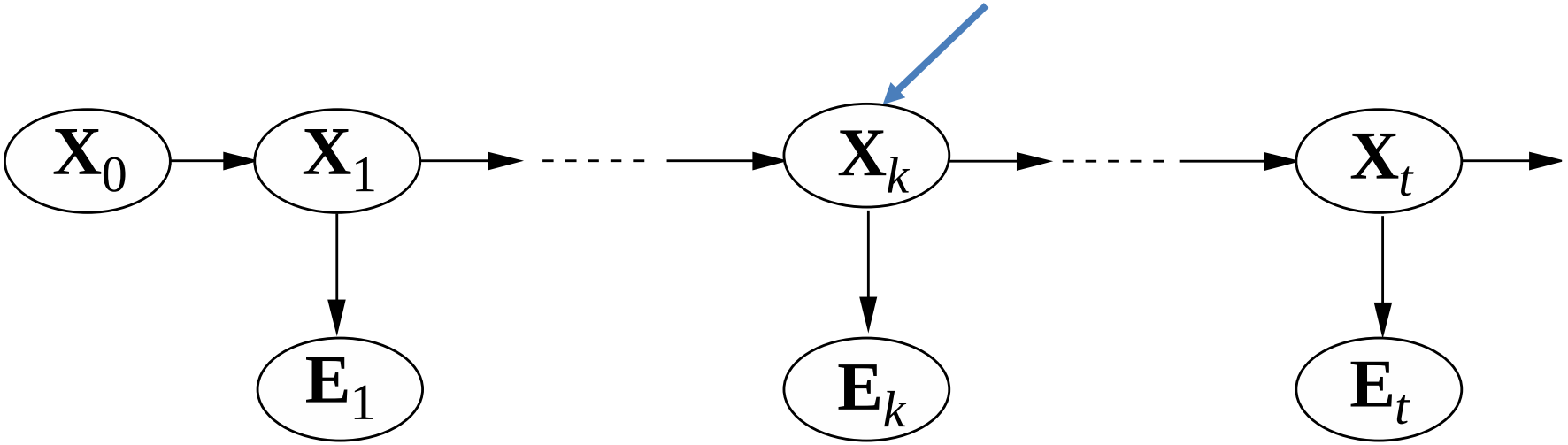
The more uncertainty there is in the transition model, the shorter will be the *mixing time* and the more the future is obscure.

Smoothing

Smoothing is the process of computing the posterior distribution of the state at **some past time** k given a complete sequence of observations up to the present t

$$\mathbf{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:t}) \qquad \text{for } 0 \leq k < t$$

The additional evidence is expected to provide more information and more accurate predictions on the past … The Bayesian network is:



Smoothing

$$\begin{aligned}
\mathbf{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:t}) &= \mathbf{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:k}, \boldsymbol{e}_{k+1:t}) && \textit{Splitting the evidence}\\
&= \alpha\, \mathbf{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:k})\, \mathbf{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k, \boldsymbol{e}_{1:k}) && \textit{Bayes's rule}\\
&= \alpha\, \mathbf{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:k})\, \mathbf{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k) && \textit{Conditional independence}\\
&= \alpha\, f_{1:k} \times b_{k+1:t} && \times \text{ is pointwise multiplication}
\end{aligned}$$

$\mathbf{P}(\boldsymbol{X}_k|\boldsymbol{e}_{1:k})$ corresponds to the *forward* message $f_{1:k}$, computed up to k, as before in filtering.

We define another *message* $b_{k+1:t} = \mathbf{P}(\boldsymbol{e}_{k+1:t}|\, \boldsymbol{X}_k)$ that can be computed by a recursive process that runs **backward** from t (next slide).

The terms in $f_{1:k}$ and $b_{k+1:t}$ can be implemented by two recursive calls:

- one running forward from 1 to k and using the *filtering equation*
- the other running backward from t to $k+1$ for computing $\mathbf{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k)$

Smoothing

$$\mathbf{P}(\boldsymbol{e}_{k+1:t}|\,\boldsymbol{X}_k) = \sum_{\boldsymbol{x}_{k+1}} \mathbf{P}(\boldsymbol{e}_{k+1:t}|\boldsymbol{X}_k, \boldsymbol{x}_{k+1})\,\mathbf{P}(\boldsymbol{x}_{k+1}|\boldsymbol{X}_k) \qquad \textit{conditioning on } \boldsymbol{X}_{k+1}$$

$$= \sum_{\boldsymbol{x}_{k+1}} \mathbf{P}(\boldsymbol{e}_{k+1:t}|\,\boldsymbol{x}_{k+1})\,\mathbf{P}(\boldsymbol{x}_{k+1}|\boldsymbol{X}_k) \qquad \textit{the evidence only depends on } \boldsymbol{x}_{k+1}$$

$$= \sum_{\boldsymbol{x}_{k+1}} \mathbf{P}(\boldsymbol{e}_{k+1}, \boldsymbol{e}_{k+2:t}|\,\boldsymbol{x}_{k+1})\,\mathbf{P}(\boldsymbol{x}_{k+1}|\boldsymbol{X}_k) \qquad \textit{splitting the evidence}$$

$$= \sum_{\boldsymbol{x}_{k+1}} P(\boldsymbol{e}_{k+1}|\,\boldsymbol{x}_{k+1})\,P(\boldsymbol{e}_{k+2:t}|\,\boldsymbol{x}_{k+1})\,\boldsymbol{P}(\boldsymbol{x}_{k+1}|\boldsymbol{X}_k) \qquad \textit{conditional independence}$$

$b_{k+1:t} = P(\boldsymbol{e}_{k+2:t}|\,\boldsymbol{x}_{k+1})$ is defined by the previous equation computing backward.

The backward phase starts with $\mathbf{P}(\boldsymbol{e}_{t+1:t}|\,\boldsymbol{X}_t) = \mathbf{1}$ **(a vector of 1's)**, because $(t+1:t)$ is an empty sequence of evidence with probability 1 of observing it.

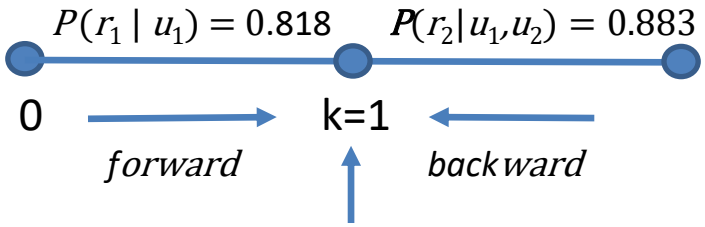
$$b_{t+1:t} = \mathbf{1}$$
$$b_{k+1:t} = \textit{Backward}(b_{k+2:t}, \boldsymbol{e}_{k+1})$$

where *Backward* implements the update defined above

Smoothing for the Umbrella World

Let's compute the smoothed estimate at time $k = 1$ for the probability of rain. Given the observations at time 1 and 2.

$$\boldsymbol{P}(R_1 \mid u_1, \underline{u}_2) = \alpha\, \boldsymbol{P}(R_1 \mid u_1) \times \boldsymbol{P}(u_2 \mid R_1)$$
$$\boldsymbol{P}(R_1 \mid u_1) = \langle 0.818, 0.182 \rangle$$



already computed in the filtering stage.

$$\boldsymbol{P}(u_2 \mid R_1) = \sum_{r_2} P(u_2 \mid r_2)\, P(U_{3:2} \mid \boldsymbol{r_2})\, \boldsymbol{P}(\boldsymbol{r_2} \mid R_1)$$

$$= (P(u_2 \mid r_2) \times \mathbf{1} \times \boldsymbol{P}(\boldsymbol{r_2} \mid R_1)) + (P(u_2 \mid \neg r_2) \times \mathbf{1} \times \boldsymbol{P}(\neg \boldsymbol{r_2} \mid R_1))$$
$$= (0.9 \times 1 \times \langle 0.7, 0.3 \rangle) + (0.2 \times 1 \times \langle 0.3, 0.7 \rangle) = \langle 0.69, 0.41 \rangle$$

$$\boldsymbol{P}(R_1 \mid u_1, \underline{u}_2) = \alpha\, \langle 0.818, 0.182 \rangle \times \langle 0.69, 0.41 \rangle \approx \langle 0.883, 0.117 \rangle$$

The **smoothed estimate** for rain on day 1 is higher than the *filtered estimate*. Rain persists.

The complexity for smoothing at a specific time k with respect to evidence $e_{1:t}$ is O(t). For smoothing a sequence O(t^2). A better linear approach exists.

forward–backward algorithm

The **forward–backward algorithm for smoothing**; computing posterior probabilities of a sequence of states given a sequence of observations:

function FORWARD-BACKWARD(**ev**, *prior*) **returns** a vector of probability distributions
  inputs: **ev**, a vector of evidence values for steps $1, \ldots, t$
    prior, the prior distribution on the initial state, $\mathbf{P}(\mathbf{X}_0)$
  local variables: **fv**, a vector of forward messages for steps $0, \ldots, t$
    b, a representation of the backward message, initially all 1s
    sv, a vector of smoothed estimates for steps $1, \ldots, t$

  $\mathbf{fv}[0] \leftarrow prior$
  for $i = 1$ **to** t **do**
    $\mathbf{fv}[i] \leftarrow \text{FORWARD}(\mathbf{fv}[i-1], \mathbf{ev}[i])$
  for $i = t$ **down to** 1 **do**
    $\mathbf{sv}[i] \leftarrow \text{NORMALIZE}(\mathbf{fv}[i] \times \mathbf{b})$
    $\mathbf{b} \leftarrow \text{BACKWARD}(\mathbf{b}, \mathbf{ev}[i])$
  return sv

Finding the most likely sequence

Suppose we observe [*true, true, false, true, true*] for the *Umbrella* variable.

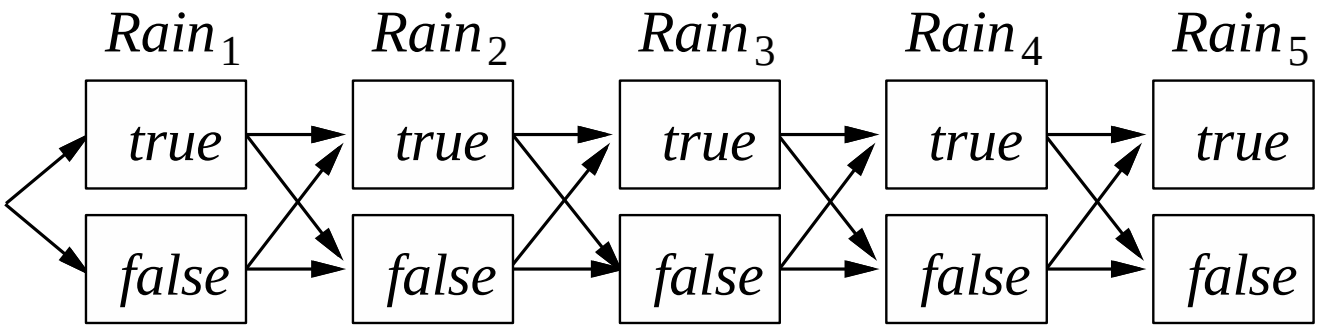
What is the most likely sequence for *Rain* given these observations?

There are 2^5 possible *Rain* sequences … Each sequence is a path through a graph whose nodes are the possible states at each time step.

We want to discover which is the one maximizing the likelihood, **in linear time**.

Likelihood of a path: product of the transition probabilities along the path and the probabilities of the given observations at each state

(a)



Trellis diagram: Square boxes are states

Finding the most likely sequence

There is a recursive relationship between the most likely path to each state x_{t+1} and most likely paths to each previous state x_t. We can write a recursive equation, similar to the one for filtering:

$$max_{x_1 \ldots xt} \boldsymbol{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, X_{t+1} \mid \boldsymbol{e}_{1:t+1}) =$$
$$= \alpha\, \boldsymbol{P}(\boldsymbol{e}_{t+1} \mid \boldsymbol{X}_{t+1})\, max_{x_t} (\boldsymbol{P}(\boldsymbol{X}_{t+1} \mid \boldsymbol{x}_t)\, max_{x_1 \ldots x_{t-1}} \boldsymbol{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{x}_t \mid \boldsymbol{e}_{1:t}))$$

The forward message in this case is

$$\boldsymbol{m}_{1:t} = max_{x_1 \ldots x_{t-1}} \boldsymbol{P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}, \boldsymbol{X}_t \mid \boldsymbol{e}_{1:t})$$

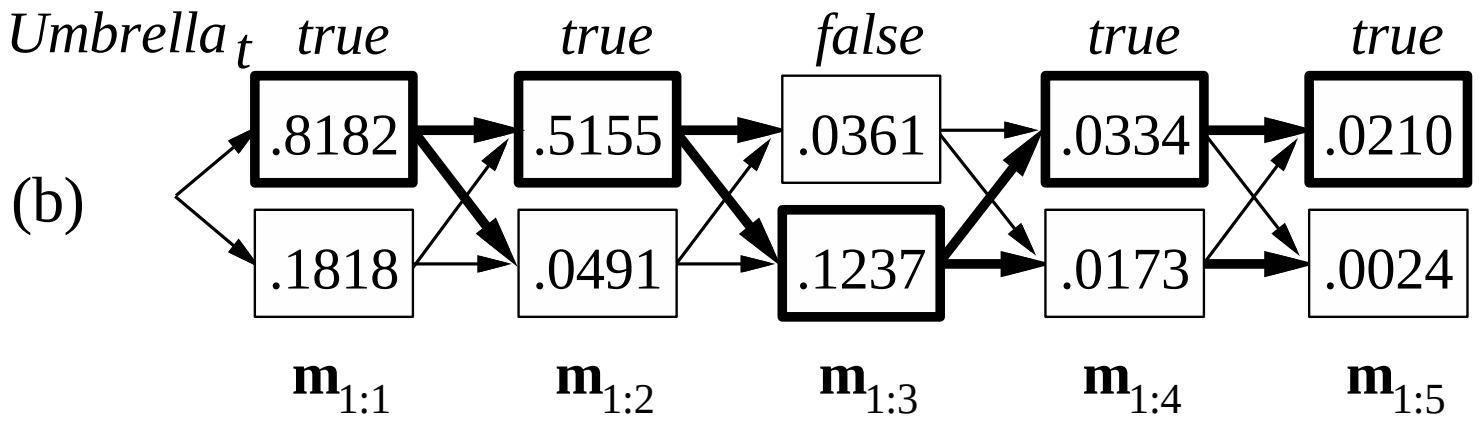
the probabilities of the best path to each state x_t; from those we can compute the probabilities of the extended paths at time t+1 and take the max.

The most likely sequence overall can be computed in one pass.

For each state, the best state that leads to it is recorded (marked as black arrows in the example next slide) so that the optimal sequence is identified by following black arrows backwards from the best final state.

This algorithm is the famous **Viterbi algorithm**, named after A. Viterbi [1967]

Finding the most likely sequence



$\mathbf{m}_{1:1} = \langle 0.8182, 0.1818 \rangle$ as computed in filtering

$\mathbf{m}_{1:2} = \langle \max[P(u_2 \mid r_2) \times 0.8182 \times P(r_2 \mid r_1), P(u_2 \mid r_2) \times 0.1818 \times P(r_2 \mid \neg r_1)],$ *rain at time 2*

$\max[P(u_2 \mid \neg r_2) \times 0.8182 \times P(\neg r_2 \mid r_1), P(u_2 \mid \neg r_2) \times 0.1818 \times P(\neg r_2 \mid \neg r_1)] \rangle$ *¬ rain at time 2*

$= \langle \max[0.9 \times 0.8182 \times 0.7, 0.9 \times 0.1818 \times 0.3], \max[0.2 \times 0.8182 \times 0.3, 0.2 \times 0.1818 \times 0.7] \rangle$

$= \langle \max[0.5155, 0.0491], \max[0.0491, 0.0255] \rangle = \langle 0.5155, 0.0491 \rangle$

$\mathbf{m}_{1:3} = P(\neg u_3 \mid R_3) \times \max_{r2}[P(R_3 \mid r_2) \times \mathbf{m}_{1:2}] = \langle 0.361, 0.1237 \rangle$

$\mathbf{m}_{1:4} = P(u_4 \mid R_4) \times \max_{r3}[P(R_4 \mid r_3) \times \mathbf{m}_{1:3}] = \langle 0.334, 0.173 \rangle$

$\mathbf{m}_{1:5} = P(u_5 \mid R_5) \times \max_{r4}[P(R_5 \mid r_4) \times \mathbf{m}_{1:4}] = \langle 0.0210, 0.0024 \rangle$

Viterbi Applications

- The original application of *Viterbi* was in telecommunications: *Viterbi decoders* are used for decoding a bitstream encoded using a technique called **convolutional code** or **trellis code**.

- In Natural Language Processing (NLP):

  - Different kinds of "sequence tagger": PoS taggers, NE taggers …
  - Dependency parsing
  - Speech recognition (speech-to-text), speech synthesis, speaker diarization (recognizing sequences from the same speaker), keyword spotting (identification of words in spoken text) …

- etc...

Hidden Markov Models

Hidden Markov model, or HMM is a temporal probabilistic model in which the state of the process is described by a **single, discrete random variable** (e.g. our *umbrella world*)

No restriction on the evidence variables. There can be many evidence variables, both discrete and continuous.

Simplified matrix algorithms

Transition model $\mathbf{P}(X_t \mid X_{t-1})$ becomes an $S \times S$ matrix $\mathbf{T}$ where:

$$T_{ij} = P(X_t = j \mid X_{t-1} = i)$$

T_{ij} is the probability of a transition from state i to state j.

$$\mathbf{T} = \mathbf{P}(X_t \mid X_{t-1}) = \begin{pmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \end{pmatrix}.$$

Umbrella World for Rain values

Hidden Markov Models

Simplified matrix algorithms (cont'd)

For mathematical convenience we place these values into an $S \times S$ diagonal **observation matrix**, $\mathbf{O}_t$, one for each time step. The ith diagonal entry of $\mathbf{O}_t$ is $P(e_t \mid X_t = i)$ and the other entries are 0. For example, on day 1 in the umbrella world of Figure 14.5, $U_1 = \mathit{true}$, and on day 3, $U_3 = \mathit{false}$, so we have

$$\mathbf{O}_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.2 \end{pmatrix}; \qquad \mathbf{O}_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.8 \end{pmatrix}.$$

Now, if we use column vectors to represent the forward and backward messages, all the computations become simple matrix–vector operations. The forward equation (14.5) becomes

$$\mathbf{f}_{1:t+1} = \alpha \mathbf{O}_{t+1} \mathbf{T}^\top \mathbf{f}_{1:t} \tag{14.12}$$

and the backward equation (14.9) becomes

$$\mathbf{b}_{k+1:t} = \mathbf{T}\mathbf{O}_{k+1}\mathbf{b}_{k+2:t}\,. \tag{14.13}$$

From these equations, we can see that the time complexity of the forward–backward algorithm (Figure 14.4) applied to a sequence of length t is $O(S^2 t)$, because each step requires multiplying an S-element vector by an $S \times S$ matrix. The space requirement is $O(St)$, because the forward pass stores t vectors of size S.

Hidden Markov Models

In HMMs, the **matrix formulation**

- Provides an elegant description of the **filtering** and **smoothing**
- reveals opportunities for improved algorithms

Online Smoothing

Online setting where smoothed estimates must be computed for earlier time slices *as new observations are continuously added to the end of the sequence*

Improved algorithms:

- simple variation on the forward–backward algorithm that allows **smoothing** to be carried out in *constant space*, independently of the length of the sequence

- **Online smoothing** with a *fixed lag* (we are smoothing at time slice t − d, where the current time is t, d is the "lag")

$$\mathbf{P}(\mathbf{X}_{t-d} \mid \mathbf{e}_{1:t})$$

Hidden Markov Models

An algorithm for **smoothing** with a fixed time lag of d steps, implemented as an online algorithm that outputs the new smoothed estimate given the observation for a new time step:

function FIXED-LAG-SMOOTHING(e_t, hmm, d) **returns** a distribution over $\mathbf{X}_{t-d}$
  inputs: e_t, the current evidence for time step t
    hmm, a hidden Markov model with $S \times S$ transition matrix $\mathbf{T}$
    d, the length of the lag for smoothing
  persistent: t, the current time, initially 1
    $\mathbf{f}$, the forward message $\mathbf{P}(X_t \mid e_{1:t})$, initially hmm.PRIOR
    $\mathbf{B}$, the d-step backward transformation matrix, initially the identity matrix
    $e_{t-d:t}$, double-ended list of evidence from $t-d$ to t, initially empty
  local variables: $\mathbf{O}_{t-d}, \mathbf{O}_t$, diagonal matrices containing the sensor model information

  add e_t to the end of $e_{t-d:t}$
  $\mathbf{O}_t \leftarrow$ diagonal matrix containing $\mathbf{P}(e_t \mid X_t)$
  if $t > d$ **then**
    $\mathbf{f} \leftarrow$ FORWARD$(\mathbf{f}, e_{t-d})$
    remove e_{t-d-1} from the beginning of $e_{t-d:t}$
    $\mathbf{O}_{t-d} \leftarrow$ diagonal matrix containing $\mathbf{P}(e_{t-d} \mid X_{t-d})$
    $\mathbf{B} \leftarrow \mathbf{O}_{t-d}^{-1}\mathbf{T}^{-1}\mathbf{B}\mathbf{T}\mathbf{O}_t$
  else $\mathbf{B} \leftarrow \mathbf{B}\mathbf{T}\mathbf{O}_t$
  $t \leftarrow t+1$
  if $t > d+1$ **then return** NORMALIZE$(\mathbf{f} \times \mathbf{B1})$ **else return** null

HMM Example: Robot Localization

The state variable X_t represents the location of the robot on the discrete grid; the domain of this variable is the set of empty squares, which we will label by the integers {1,...,S}.

Let NEIGHBORS(i) be the set of empty squares that are adjacent to i and let N(i) be the size of that set.

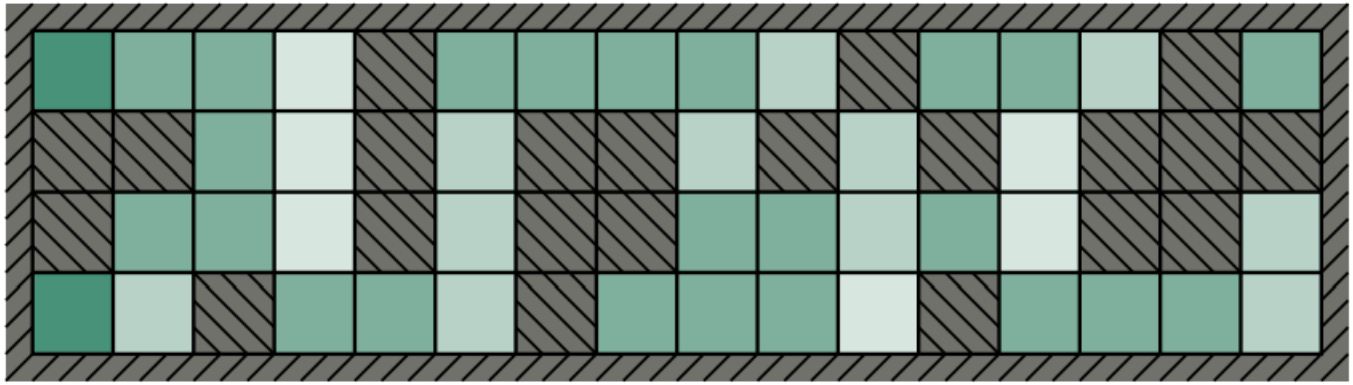
Then the **transition model** for the Move action says that the robot is equally likely to end up at any neighboring square:

$$P(X_{t+1}=j \mid X_t=i) = \mathbf{T}_{ij} = \begin{cases} 1/N(i) \text{ if } j \in \text{NEIGHBORS}(i) \\ 0 \text{ otherwise.} \end{cases}$$

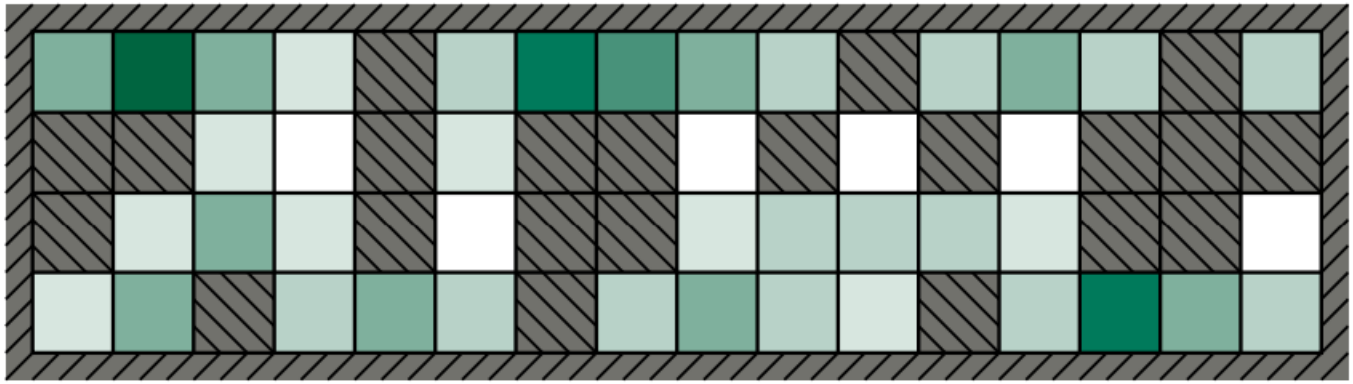
We don't know where the robot starts, so we will assume a uniform distribution over all the squares; that is, $P(X_0=i) = 1/S$.

The sensor variable E_t has 16 possible values, each a four-bit sequence giving the **presence or absence of an obstacle** in each of the compass directions NESW. For example, "1010" means that the north and south sensors report an obstacle and the east and west do not.

HMM Example: Robot Localization



(a) Posterior distribution over robot location after $E_1 = 1011$



(b) Posterior distribution over robot location after $E_1 = 1011, E_2 = 1010$

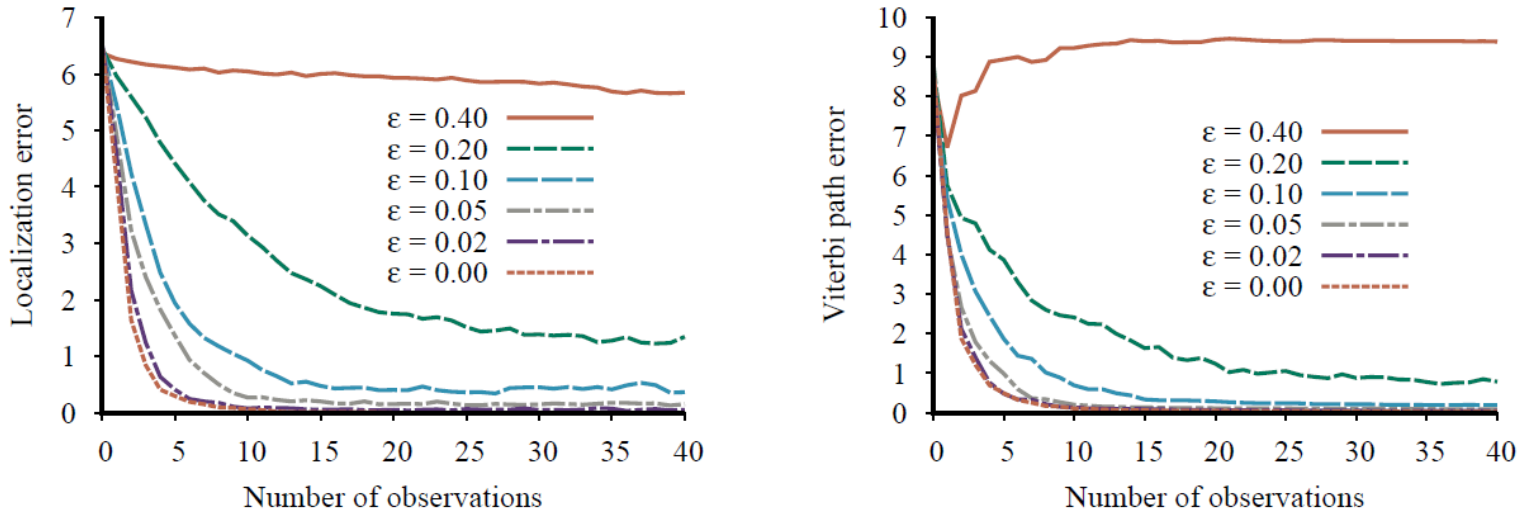
Figure 14.7 Posterior distribution over robot location: (a) after one observation $E_1 = 1011$ (i.e., obstacles to the north, south, and west); (b) after a random move to an adjacent location and a second observation $E_2 = 1010$ (i.e., obstacles to the north and south). The darkness of each square corresponds to the probability that the robot is at that location. The sensor error rate for each bit is $\epsilon = 0.2$.

HMM Example: Robot Localization

In this example we could use:

- **Filtering:** to estimate its current robot location
- **Smoothing**: to work out where it was at any given past time—for example, where it began at time 0.
- **Viterbi algorithm:** to work out the most likely path it has taken to get where it is now.

HMM Example: Robot Localization



Performance of **HMM localization** as a function of the length of the observation sequence for various different values of the sensor error probability *E*; data averaged over 400 runs.

(a) The localization error, defined as the Manhattan distance from the true location.
(b) The Viterbi path error, defined as the average Manhattan distance of states on the Viterbi path from corresponding states on the true path

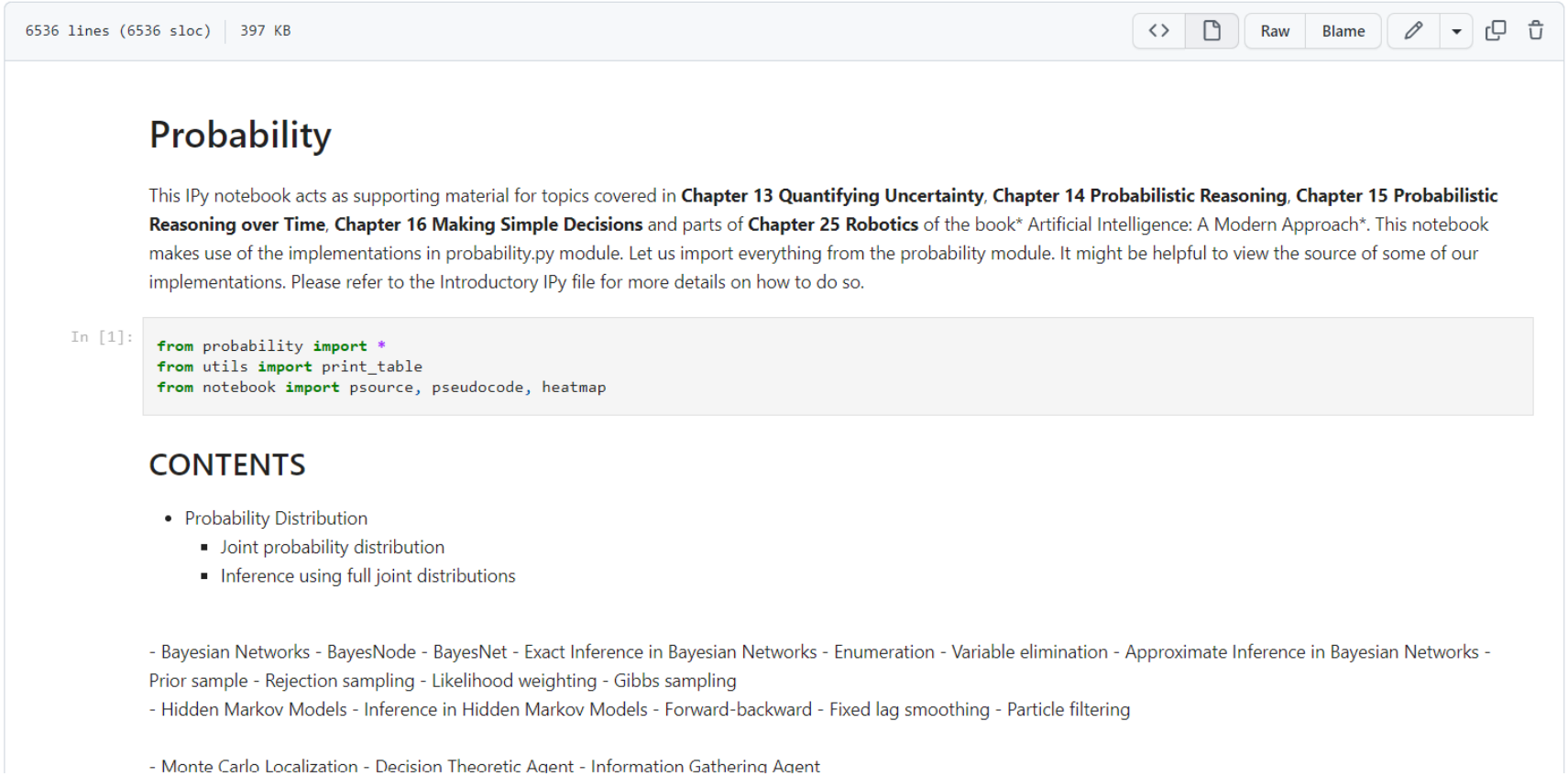
HMMs Applications

HMMs have been applied successfully to a wide variety of fields such as:

- Thermodynamics
- Statistical mechanics
- Physics
- Chemistry
- Economics
- Finance
- Signal processing
- Information theory
- Pattern recognition
  - speech, handwriting, gesture recognition, part-of-speech tagging, partial discharges ,bioinformatics.

- *…your project?*

AIMA-python - “Probability.ipynb”



6536 lines (6536 sloc) | 397 KB

Raw Blame

Probability

This IPy notebook acts as supporting material for topics covered in **Chapter 13 Quantifying Uncertainty**, **Chapter 14 Probabilistic Reasoning**, **Chapter 15 Probabilistic Reasoning over Time**, **Chapter 16 Making Simple Decisions** and parts of **Chapter 25 Robotics** of the book* Artificial Intelligence: A Modern Approach*. This notebook makes use of the implementations in probability.py module. Let us import everything from the probability module. It might be helpful to view the source of some of our implementations. Please refer to the Introductory IPy file for more details on how to do so.

In [1]:

```
from probability import *
from utils import print_table
from notebook import psource, pseudocode, heatmap
```

CONTENTS

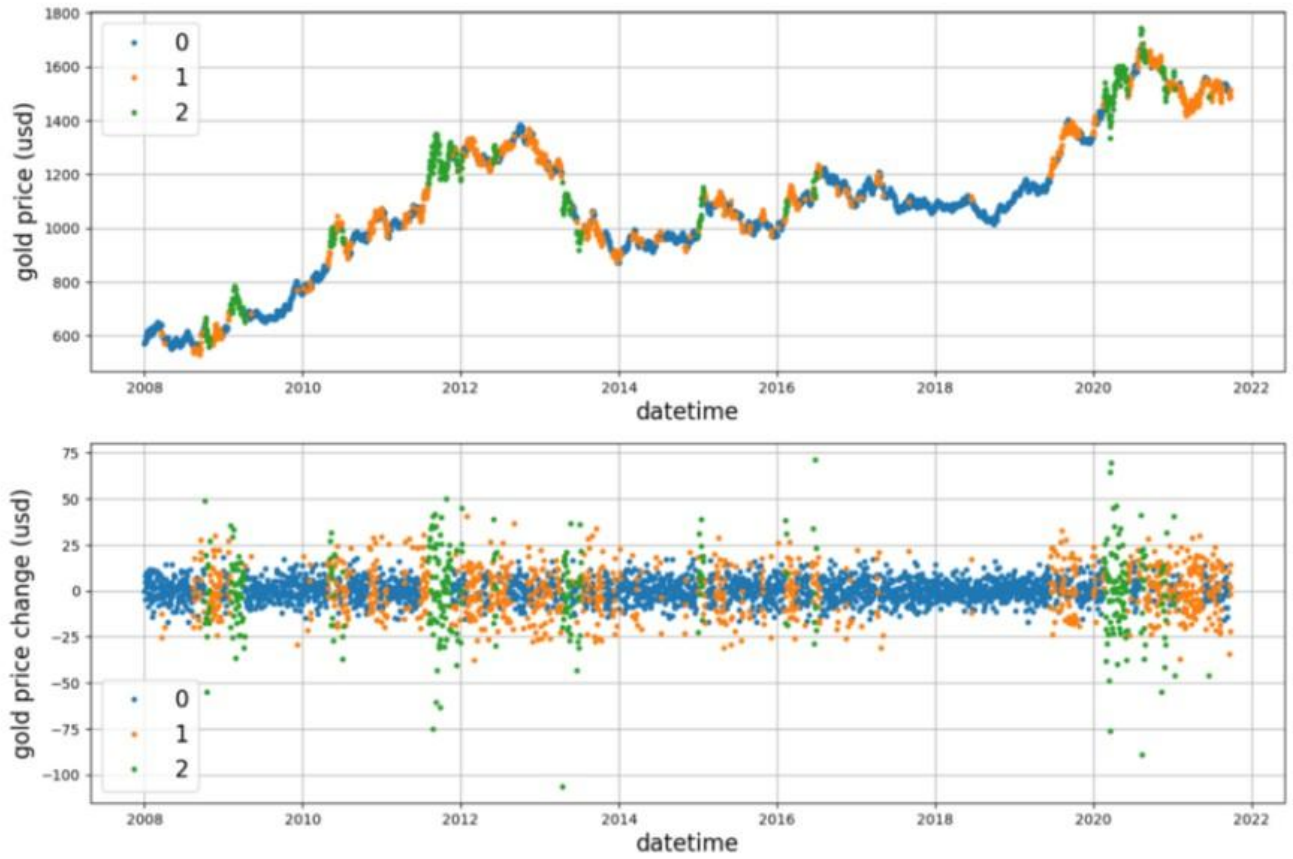
- Probability Distribution
  - Joint probability distribution
  - Inference using full joint distributions

- Bayesian Networks - BayesNode - BayesNet - Exact Inference in Bayesian Networks - Enumeration - Variable elimination - Approximate Inference in Bayesian Networks - Prior sample - Rejection sampling - Likelihood weighting - Gibbs sampling
- Hidden Markov Models - Inference in Hidden Markov Models - Forward-backward - Fixed lag smoothing - Particle filtering

- Monte Carlo Localization - Decision Theoretic Agent - Information Gathering Agent

[aima-python/probability.ipynb at master · aimacode/aima-python (github.com)]

Market Volatility with HMMlearn



Market volatility as modeled using a Gaussian emissions Hidden Markov Model. Blue/state 0 — low volatility, orange/state 1— medium volatility, green/state 2 — high volatility. Image created by the author.

[Hidden Markov Models with Python. Modelling Sequential Data… | by Y. Natsume]

Summary

- The changing state of the world is handled by using a set of **random variables** to represent the state **at each point in time**.
- Representations can be designed to (roughly) satisfy the **Markov property**, so that **the future is independent of the past given the present**. Combined with the assumption that the process is time-homogeneous, this greatly simplifies the representation.
- A **temporal probability model** can be thought of as containing a transition model describing the state evolution and a sensor model describing the observation process.
- The principal inference tasks in temporal models are **filtering** (state estimation), **prediction**, **smoothing**, and computing the **most likely explanation**.
- **Hidden Markov Models** are a family of "simple" markov models with **single discrete variables** but extensively used in practice to solve real-world problems

In the next lecture...

- Properties of Multiagent Environments
- Non-Cooperative Game Theory
- Cooperative Game Theory
- Making Collective Decisions