Unit-3

Probability Reasoning with Time:

Agents in partially observable environments must be able to keep track of the current state, to the extent that their sensors allow. An agent maintains a **belief state** that represents which states of the world are currently possible. From the belief state and a **transition model**, the agent can predict how the world might evolve in the next time step. From the percepts observed and a **sensor model**, the agent can update the belief state. A changing world is modeled using a variable for each aspect of the world state *at each point in time*. The transition and sensor models may be uncertain: the transition model describes the probability distribution of the variables at time t, given the state of the world at past times, while the sensor model describes the probability of each percept at time t, given the current state of the world.

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

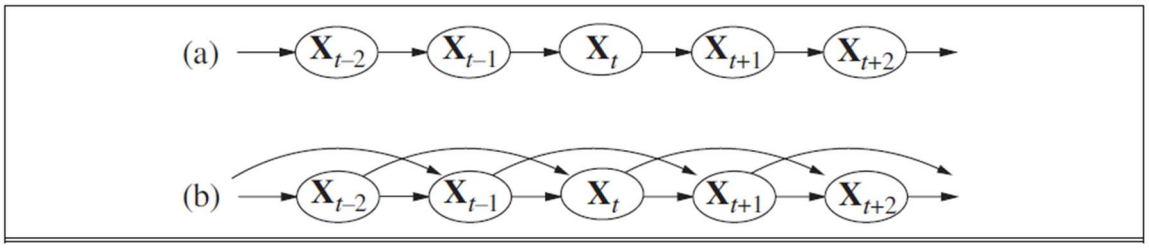
Consider a slightly different problem: treating a diabetic patient. As in the case of car repair, we have evidence such as recent insulin doses, food intake, blood sugar measurements, and other physical signs. The task is to assess the current state of the patient, including the actual blood sugar level and insulin level. Given this information, we can make a decision about the patient's food intake and insulin dose. Unlike the case of car repair, here the *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.

We view the world as a series of snapshots, or **time slices**, each of which contains a set of random variables, some observable and some not. For simplicity, we will assume that the same subset of variables is observable in each time slice. We will use $\mathbf{X}_t$ to denote the set of state variables at time t, which are assumed to be unobservable, and $\mathbf{E}_t$ to denote the set of observable evidence variables. The observation at time t is $\mathbf{E}_t = \mathbf{e}_t$ for some set of values **e**t.

Consider the following example: 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 Umbrella t or U_t for short, and the set **X**t contains a single state variable Raint or R_t for short.

The interval between time slices also depends on the problem. For diabetes monitoring, a suitable interval might be an hour rather than a day. In this chapter we assume the interval between slices is fixed, so we can label times by integers. We will assume that the state sequence starts at t=0; for various uninteresting reasons, we will assume that evidence starts arriving at t=1 rather than t=0. Hence, our umbrella world is represented by state variables R0, R1, R2, . . . and evidence variables U1, U2, ... We will use the notation a:b to denote the sequence of integers from a to b (inclusive), and the notation **X**a:b to denote the set of variables from **X**a to **X**b. For example, U1:3 corresponds to the variables U1, U2, U3.

The transition model specifies the probability distribution over the latest state variables, given the previous values, that is, $\mathbf{P}(\mathbf{X}t \mid \mathbf{X}0{:}t-1)$. Now we face a problem: the set $\mathbf{X}0{:}t-1$ is MARKOV unbounded in size as t increases. We solve the problem by making a **Markov assumption**—that the current state depends on only a *finite fixed number* of previous states. Processes satisfying this assumption were first studied in depth by the Russian statistician Andrei Markov (1856–1922) and are called **Markov processes** or **Markov chains**.



The simplest is the **first-order Markov process**, in which the current state depends only MARKOV PROCESS on the previous state and not on any earlier states. In other words, a state provides enough information to make the future conditionally independent of the past, and we have

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

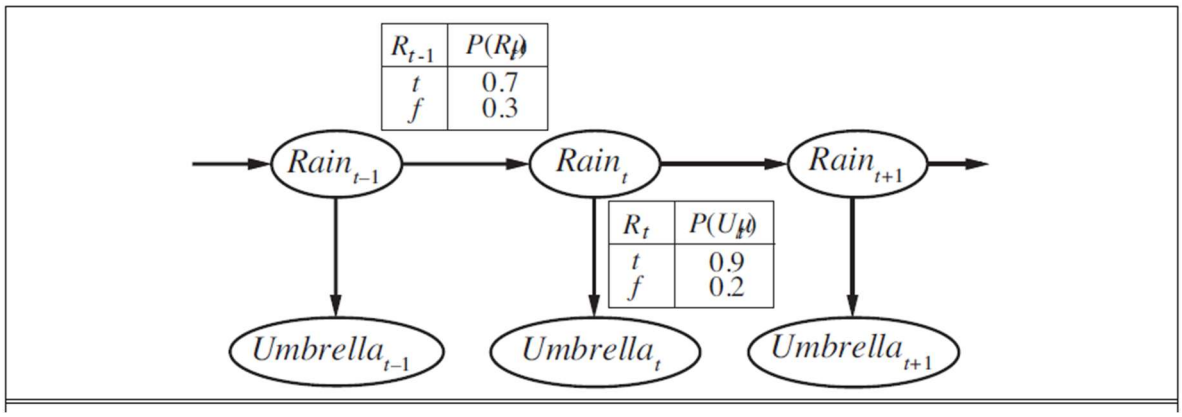
A **stationary process**— is, a process of change that is governed by laws that do not themselves change over time.

In the umbrella world, the conditional probability of rain, $\mathbf{P}(\mathrm{R}t \mid \mathrm{R}t\text{-}1)$, is the same for all t, and we only have to specify one conditional probability table.

The evidence variables $\mathbf{E}_t$ *could* depend on previous variables as well as the current state variables, generate the current sensor values. we make a **sensor Markov assumption** as follows:

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

$\mathbf{P}(\mathbf{E}t \mid \mathbf{X}t)$ is our sensor model



the arrows go from the actual state of the world to sensor values because the state of the world *causes* the sensors to take on particular values: the rain *causes* the umbrella to appear.

The prior probability distribution at time 0, $\mathbf{P}(\mathbf{X}_0)$. With that, we have a specification of the complete joint distribution over all the variables, using Equation

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

The three terms on the right-hand side are the initial state model $\mathbf{P}(\mathbf{X}_0)$, the transition model $\mathbf{P}(\mathbf{X}_i \mid \mathbf{X}_{i-1})$, and the sensor model $\mathbf{P}(\mathbf{E}_i \mid \mathbf{X}_i)$.

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.

First-order Markov process—the probability of rain is assumed to depend only on whether it rained the previous day.

There are two ways to improve the accuracy of the approximation:

1. Increasing the order of the Markov process model. For example, we could make a second-order model by adding Rain_{t-2} as a parent of Rain_t.
2. Increasing the set of state variables For example, we could add Seasont to allow us to incorporate historical records of rainy seasons, or we could add Temperature $_t$, Humidityt and Pressuret to allow us to use a physical model of rainy conditions.

Consider, for example, the problem of tracking a robot wandering randomly on the X–Y plane. One might propose that the position and velocity are a sufficient set of state variables: one can simply use Newton's laws to calculate the new position, and the velocity may change unpredictably. If the robot is battery-powered, however, then battery exhaustion would tend to have a systematic effect on the change in velocity. Because this in turn depends on how much power was used by all previous maneuvers, the Markov property is violated. We can restore the Markov property by including the charge level Battery_t as one of the state variables that make up $\mathbf{X}_t$. This helps in predicting the motion of the robot, but in turn requires a model for predicting Battery_t from Battery_{t-1} and the velocity. In some cases, that can be done reliably, but more often we find that error accumulates over time. In that case, accuracy can be improved by *adding a new sensor* for the battery level.

Inferences in Temporal model:

Filtering: The task of computing the **belief state**—the posterior distribution over the most recent state—given all evidence to date. Filtering is also called **stateestimation**. $\mathbf{P}(\mathbf{X}_t \mid \mathbf{e}_{1:t})$. In the umbrella example, this would mean computing the probability of rain today, given all the observations of the umbrella carrier made so far. Filtering is what a rational agent does to keep track of the current state so that rational decisions can be made. It turns out that an almost identical calculation provides the likelihood of the evidence sequence, $\mathsf{P}(\mathbf{e}_{1:t})$.

$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t+1}) = \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t}, \mathbf{e}_{t+1})$ (dividing up the evidence)

$= \alpha \ \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}, \mathbf{e}_{1:t}) \, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ (using Bayes' rule)

$= \alpha \ \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \, \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{e}_{1:t})$ (by the sensor Markov assumption)

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

$$= \alpha\, \mathbf{P}(\mathbf{e}_{t+1} \mid \mathbf{X}_{t+1}) \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t) P(\mathbf{x}_t \mid \mathbf{e}_{1:t}) \quad \text{(Markov assumption)}.$$

Prediction: This is the task of computing the posterior distribution over the *future* state, given all evidence to date. That is, we wish to compute $\mathbf{P}(\mathbf{X}_{t+k} \mid \mathbf{e}_{1:t})$ for some $k > 0$. In the umbrella example, this might mean computing the probability of rain three days from now, given all the observations to date. Prediction is useful for evaluating possible courses of action based on their expected outcomes.

$$\mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{e}_{1:t}) = \sum_{\mathbf{x}_{t+k}} \mathbf{P}(\mathbf{X}_{t+k+1} \mid \mathbf{x}_{t+k}) P(\mathbf{x}_{t+k} \mid \mathbf{e}_{1:t}) \ .$$

Smoothing: This is the task SMOOTHING of computing the posterior distribution over a *past* state, given all evidence up to the present. That is, we wish to compute $\mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:t})$ for some k such that $0 \leq k < t$. In the umbrella example, it might mean computing the probability that it rained last Wednesday, given all the observations of the umbrella carrier made up to today. Smoothing provides a better estimate of the state than was available at the time, because it incorporates more evidence

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

$= \alpha\ \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}) \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{e}_{1:k})$ (using Bayes' rule)

$= \alpha\ \mathbf{P}(\mathbf{X}_k \mid \mathbf{e}_{1:k}) \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$ (using conditional independence)

$= \alpha\ \mathbf{f}_{1:k} \times \mathbf{b}_{k+1:t}$

$\mathbf{b}_{k+1:t} = \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$

$$\begin{aligned}
\mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k) &= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k, \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{(conditioning on } \mathbf{X}_{k+1}\text{)} \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} \mid \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \quad \text{(by conditional independence)} \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} \mid \mathbf{x}_{k+1}) P(\mathbf{e}_{k+2:t} \mid \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} \mid \mathbf{X}_k) \ , \qquad (15.9)
\end{aligned}$$

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.

compute $\text{argmax}_{\mathbf{x}_{1:t}} \mathbf{P}(\mathbf{x}_{1:t} \mid \mathbf{e}_{1:t})$.

For example, if the umbrella appears on each of the first three days and is absent on the fourth, then the most likely explanation is that it rained on the first three days and did not rain on the fourth. Algorithms for this task are useful in many applications, including speech recognition—where the aim is to find the most likely sequence of words, given a series of sounds—and the reconstruction of

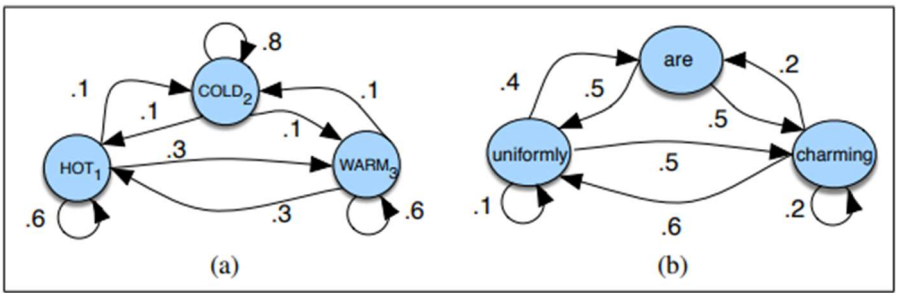
bit strings transmitted over a noisy channel.

Learning: The transition and sensor models, if not yet known, can be learned from observations. Just as with static Bayesian networks, dynamic Bayes net learning can be done as a by-product of inference. Inference provides an estimate of what transitions actually occurred and of what states generated the sensor readings, and these estimates can be used to update the models. The updated models provide new estimates, and the process iterates to convergence. The overall process is an instance of the expectation maximization or **EM algorithm**.

Learning requires smoothing, rather than filtering, because smoothing provides better estimates of the states of the process. Learning with filtering can fail to converge correctly; consider, for example, the problem of learning to solve murders: unless you are an eyewitness, smoothing is *always* required to infer what happened at the murder scene from the observable variables.

Hidden Markov Model

The HMM is based on augmenting the Markov chain. A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set. These sets can be words, or tags, or symbols representing anything, like the weather. A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. The states before the current state have no impact on the future except via the current state.



Consider a sequence of state variables $q_1,q_2,...,q_i$. A Markov model embodies the Markov assumption on the probabilities of this sequence: that Markov assumption when predicting the future, the past doesn't matter, only the present.

Markov Assumption:

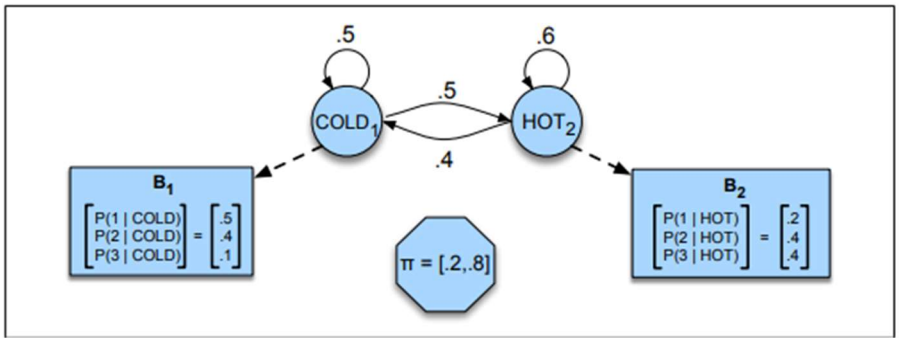
$$P(q_i = a \mid q_1 ... q_{i-1}) = P(q_i = a \mid q_{i-1})$$

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{N} \pi_i = 1$

A Markov chain is useful when we need to compute a probability for a sequence of observable events. the events we are interested in are hidden. A hidden Markov model (HMM) allows us to talk about both observed events Hidden Markov model and hidden events that we think of as causal factors in our probabilistic model.

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Given a sequence of observations O (each an integer representing the number of ice creams eaten on a given day) find the 'hidden' sequence Q of weather states (H or C) which caused Jason to eat the ice cream.



hidden Markov models should be characterized by three fundamental problems

Problem 1 (Likelihood):	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Problem 2 (Decoding):	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Problem 3 (Learning):	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

Likelihood Computation: The Forward Algorithm

Computing Likelihood: Given an HMM λ = (A,B) and an observation sequence O, determine the likelihood P(O|λ). Where λ is set of hidden states.

The surface observations are the same as the hidden events, we could compute the probability of 3 1 3 just by following the states labeled 3 1 3 and multiplying the probabilities along the arcs.

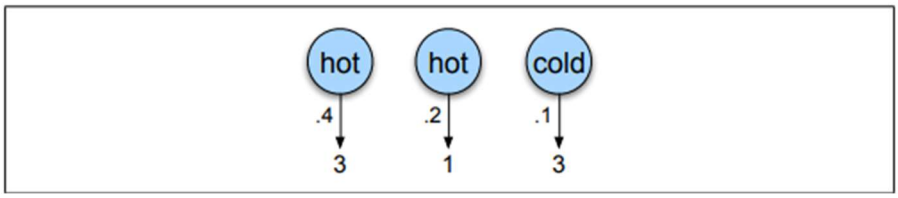
First, recall that for hidden Markov models, each hidden state produces only a single observation. Thus, the sequence of hidden states and the sequence of observations have the same length

Given this one-to-one mapping and the Markov assumptions for a particular hidden state sequence $Q = q_0,q_1,q_2,...,q_T$ and an observation sequence $O = o_1,o_2,...,o_T$, the likelihood of the observation sequence is

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

The computation of the forward probability for our ice-cream observation 3 1 3 from one possible hidden state sequence hot hot cold

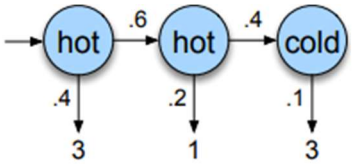
P(3 1 3|hot hot cold) = P(3|hot)×P(1|hot)×P(3|cold)



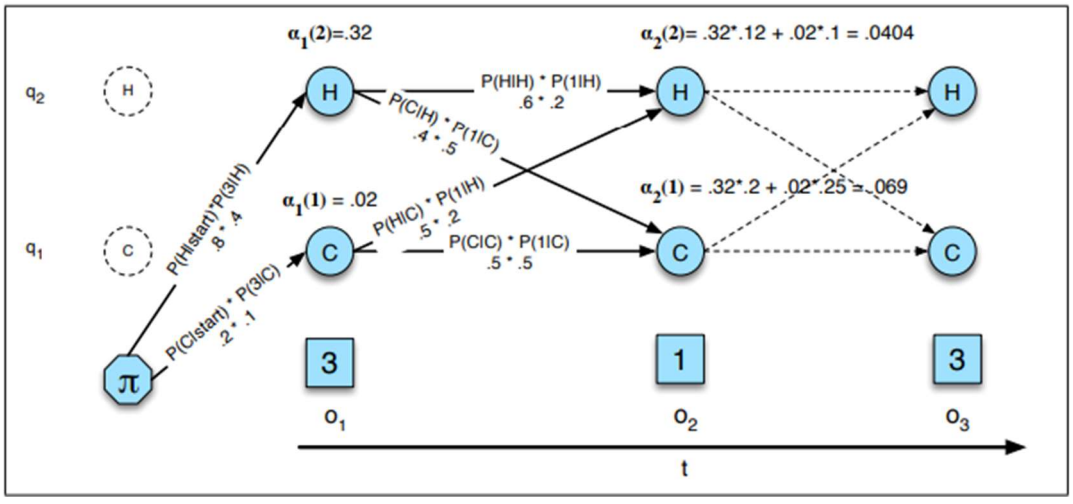
compute the probability of ice-cream events 3 1 3 instead by summing over all possible weather sequences, weighted by their probability. First, let's compute the joint probability of being in a particular weather sequence Q and generating a particular sequence O of ice-cream events.

$$P(O,Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \times \prod_{i=1}^{T} P(q_i|q_{i-1})$$

P(3 1 3,hot hot cold) = P(hot|start)×P(hot|hot)×P(cold|hot) ×P(3|hot)×P(1|hot)×P(3|cold)



P(3 1 3) = P(3 1 3, cold cold cold) +P(3 1 3, cold cold hot) +P(3 1 3,hot hot cold) +...



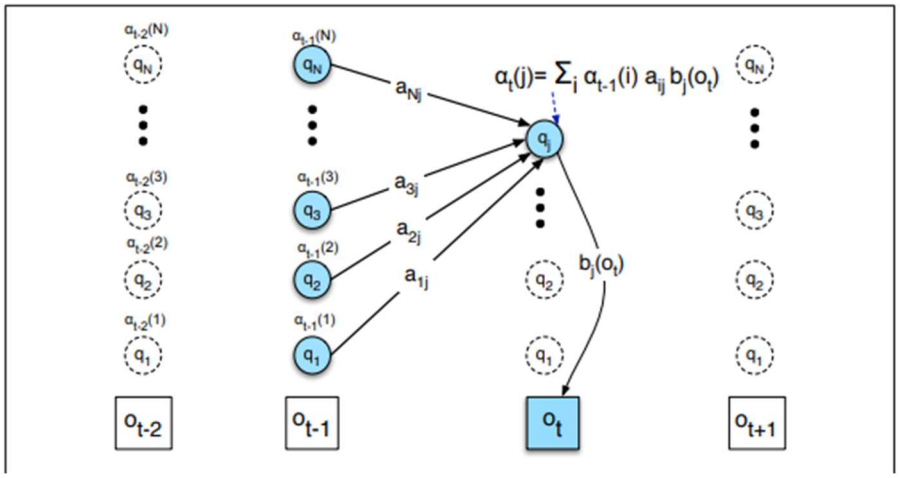
Each cell of the forward algorithm trellis $\alpha_t(j)$ represents the probability of being in state j after seeing the first t observations, given the automaton λ. The value of each cell $\alpha_t(j)$ is computed by summing over the probabilities of every path that could lead us to this cell. Formally, each cell expresses the following probability:

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j | \lambda)$$

q_t=j means "the t th state in the sequence of states is state j". We compute this probability $\alpha_t(j)$ by summing over the extensions of all the paths that lead to the current cell. For a given state q_j at time t, the value $\alpha_t(j)$ is computed as

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

$\alpha_{t-1}(i)$	the **previous forward path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j



function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward*[N,T]
for each state s **from** 1 **to** N **do** ; initialization step
  $forward[s,1] \leftarrow \pi_s * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$$; termination step
return *forwardprob*

Decoding: The Viterbi Algorithm:

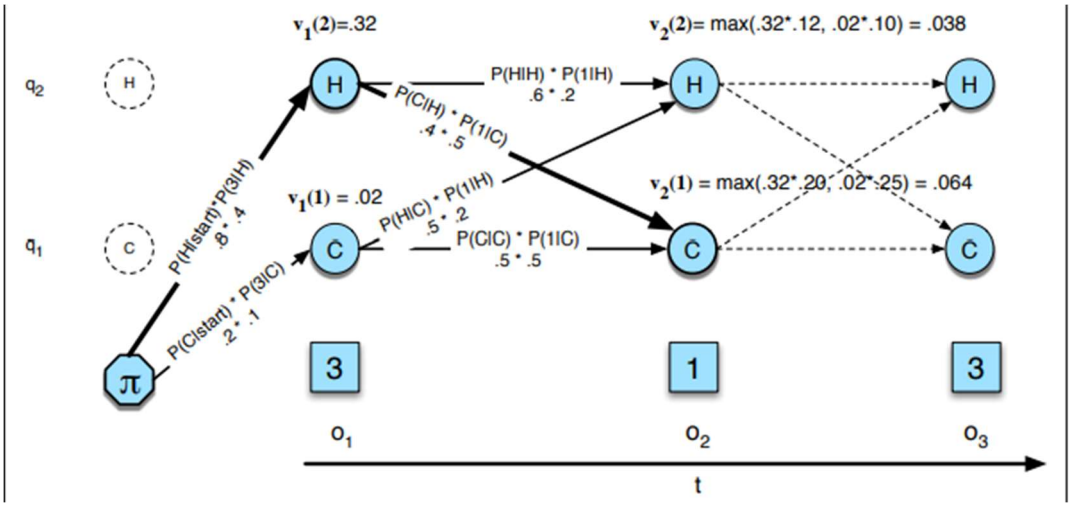
The task of determining which sequence of variables is the underlying source of some sequence of decoding observations is called the decoding task.

In the ice-cream domain, given a sequence of ice-cream observations 3 1 3 and an HMM, the task of the decoder is to find the best hidden weather sequence (H H H)

Decoding: Given as input an HMM λ = (A,B) and a sequence of observations $O = o_1,o_2,...,o_T$, find the most probable sequence of states $Q = q_1q_2q_3 ...q_T$.

Find the best sequence as follows: For each possible hidden state sequence (HHH, HHC, HCH, etc.), we could run the forward algorithm and compute the likelihood of the observation sequence given that hidden state sequence. The hidden state sequence with the maximum observation likelihood.

The most common decoding algorithms for HMMs is the Viterbi algorithm. Like the forward algorithm, Viterbi is a kind of dynamic programming Viterbi algorithm that makes uses of a dynamic programming trellis.



The Viterbi trellis for computing the best hidden state sequence for the observation sequence 3 1 3. The idea is to process the observation sequence left to right, filling out the trellis. Each cell of the trellis, $v_t(j)$, represents the probability that the HMM is in state j after seeing the first t observations and passing through the most probable state sequence $q_1,...,q_{t-1}$.

$$v_t(j) = \max q_1,...,q_{t-1} P(q_1...q_{t-1},o_1,o_2 ...o_t ,q_t = j|\lambda)$$

we compute the Viterbi probability by taking the most probable of the extensions of the paths that lead to the current cell. For a given state q_j at time t, the value $v_t(j)$ is computed as

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

function VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state *s* **from** 1 **to** *N* **do** ; initialization step
viterbi[*s*,1] $\leftarrow \pi_s * b_s(o_1)$
backpointer[s,1] $\leftarrow 0$
for each time step *t* **from** 2 **to** *T* **do** ; recursion step
for each state *s* **from** 1 **to** *N* **do**
viterbi[s,t] $\leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
backpointer[s,t] $\leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
bestpathprob $\leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
bestpathpointer $\leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

HMM Training: The Forward-Backward Algorithm:

Learning: Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B.

The standard algorithm for HMM training is the forward-backward, or Baum-Welch algorithm

considering the much simpler case of training a fully visible Markov model, we're know both the temperature and the ice cream count for every day. That is, imagine we see the following set of input observations and magically knew the aligned hidden state sequences:

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

This would easily allow us to compute the HMM parameters just by maximum likelihood estimation from the training data. First, we can compute π from the count of the 3 initial hidden states: 4

$$\pi_h = 1/3 \; \pi_c = 2/3$$

A Matrix p(hot|hot) = 2/3 p(cold|hot) = 1/3
p(cold|cold) = 2/3 p(hot|cold) = 1/3

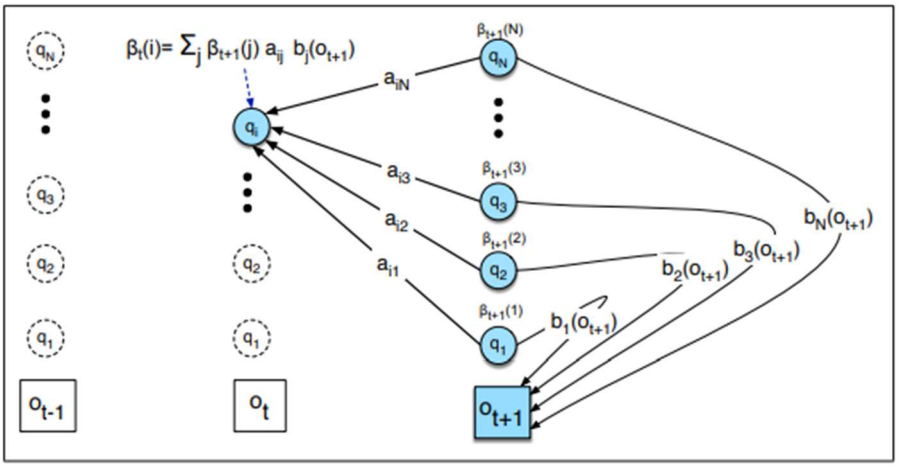
and the B matrix:

P(1|hot) = 0/4 = 0
p(1|cold) = 3/5 = .6
P(2|hot) = 1/4 = .25
p(2|cold = 2/5 = .4
P(3|hot) = 3/4 = .75
p(3|cold) = 0

The Baum-Welch algorithm solves this by iteratively estimating the counts. We will start with an estimate for the transition and observation probabilities and then use these estimated probabilities to derive better and better probabilities. computing the forward probability for an observation and then dividing that probability mass among all the different paths that contributed to this forward probability. we need to define a useful probability related to the forward probability and called the

backward probability. The backward probability β is the probability of seeing the observations from time t +1 to the end, given that we are in state i at time t (and given the automaton λ):

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots o_T \mid q_t = i, \lambda)$$



$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)} \qquad \xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j) \beta_t(j)}$$

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1\, s.t. O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \qquad \gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}$$

function FORWARD-BACKWARD(*observations* of len *T*, *output vocabulary V*, *hidden state set Q*) **returns** *HMM=(A,B)*

initialize *A* and *B*
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return *A*, *B*