Chapter 2

Introduction to State Estimation

Reading

1. Barfoot, Chapter 2.1-2.2
2. Thrun, Chapter 2
3. Russell Chapter 15.1-15.3

2.1 A review of probability

Probability is a very useful construct to reason about real systems which we cannot model fully at all scales. It is a fundamental part of robotics. No matter how sophisticated your camera, it will have noise in how it measures the real world around it. No matter how good your model for a motor is, there will be modeled effects which make it move a little differently than how you would expect. We begin with a quick review of probability, you can read more at many sources, e.g., MIT's OCW.

An experiment is a procedure which can be repeated infinitely and has a well-defined set of possible outcomes, e.g., the toss of a coin or the roll of dice. The outcome itself need not always be deterministic, e.g., depending upon your experiment, the coin may come up heads or tails. We call the set Ω the *sample space*, it is the set of all possible outcomes of an experiment. For two coins, this set would be

$$\Omega = \{HH, HT, TH, TT\}.$$

We want to pick this set to be right granularity to answer relevant questions, e.g., it is correct but not useful for Ω to be position of all the molecules in the coin. After every experiment, in this case tossing the two coins once each, we obtain an event, it is a subset *event* $A \subset \Omega$ from the sample space.

$$A = \{HH\}.$$

Probability theory is a mathematical framework that allows us to reason about phenomena or experiments whose outcome is uncertain. Probability of an event

$$\mathrm{P}(A)$$

is a function that maps each event A to a number between 0 and 1: closer to 1 this number, stronger our belief that the outcome of the experiment is going to be A.

ℹ What is an Axiom?

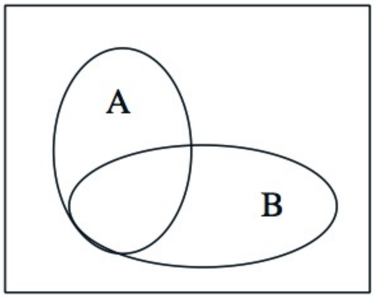
Axioms Probability is formalized using a set of three basic axioms that are intuitive and yet very powerful, they are known a Kolmogorov's axioms. They are

- Non-negativity: $\mathrm{P}(A) \geq 0$
- Normalization: $\mathrm{P}(\Omega) = 1$
- Additivity: If two events A, B are such that $A \cap B = \emptyset$, then

$$\mathrm{P}(A \cup B) = \mathrm{P}(A) + \mathrm{P}(B).$$

You can use these axioms to say things like $\mathrm{P}(\emptyset) = 0$, $\mathrm{P}(A^c) = 1 - \mathrm{P}(A)$, or if $A \subseteq B$ then $\mathrm{P}(A) \leq \mathrm{P}(B)$.

ℹ



Conditioning on events Conditioning helps us answer questions like

$$\mathrm{P}(A \mid B) := \text{probability of} A \text{ given that } B \text{ occurred}.$$

Effectively, the sample space has now shrunk from Ω to the event B. It would be silly to have a null sample-space, so let's say that $\mathrm{P}(B) \neq 0$. We define conditional probability as

$$\mathrm{P}(A \mid B) = \frac{\mathrm{P}(A \cap B)}{\mathrm{P}(B)}; \tag{2.1}$$

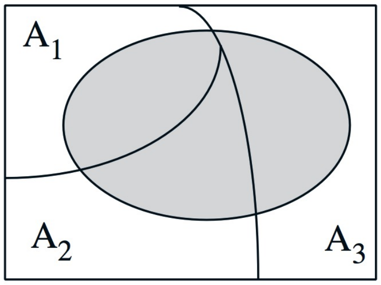
the probability is undefined if $\mathrm{P}(B) = 0$. Using this definition, we can compute the probability of events like "what is the probability of rolling a 2 on a die given that an even number was rolled".

We can use this trick to get the law of total probability: if a finite number of events $\{A_i\}$ form a partition of Ω, i.e.,

$$A_i \cap A_j = \emptyset \; \forall i, j, \text{ and } \bigcup_i A_i = \Omega$$

$$\mathrm{P}(B) = \sum_i \mathrm{P}(B \mid A_i) \; \mathrm{P}(A_i). \tag{2.2}$$

ℹ Partitioning the sample space



Bayes' rule Imagine that instead of someone telling us that the conditioning event actually happened, we simply had a belief

$$\mathrm{P}(A_i)$$

about the possibility of such events $\{A_i\}$. For each of A_i, we can compute the conditional probability $\mathrm{P}(B \mid A_i)$ using (2.1). Say we run our experiment and observe that B occurred, how would our belief on the events A_i change? In other words, we wish to compute

$$\mathrm{P}(A_i \mid B).$$

This is the subject of Bayes' rule.

$$\begin{aligned} \mathrm{P}(A_i \mid B) &= \frac{\mathrm{P}(A_i \ \cap B)}{\mathrm{P}(B)} \\ &= \frac{\mathrm{P}(A_i)\,\mathrm{P}(B|A_i)}{\mathrm{P}(B)} \\ &= \frac{\mathrm{P}(A_i)\,\mathrm{P}(B|A_i)}{\sum_i \mathrm{P}(A_j)\ \mathrm{P}(B \mid A_j)}. \end{aligned} \tag{2.3}$$

The Bayes' rule naturally leads to the concept of independent events. Two events $A, B \subset \Omega$ are independent if observing one does not give us any information about the other

$$\mathrm{P}(A \cap B) = \mathrm{P}(A)\,\mathrm{P}(B). \tag{2.4}$$

This is different from disjoint events. Disjoint events never co-occur, i.e., observing one tells us that the other one *did not* occur.

Probability for experiments with real-valued outcomes We need some more work in defining probability for events with real-valued outcomes. The sample space is easy enough to understand, e.g., $\Omega = [0, 1]$ for your score at the end of this course. We however run into difficulties if we define the probability of general subsets of Ω in terms of the probabilities of elementary outcomes (*elements* of Ω). For instance, if we wish to model all elements $\omega \in \Omega$ to be equally likely, we are forced to assign each element ω a probability of zero (to be consistent with the second axiom of probability). This is not very helpful in determining the probability of the score being 0.9. If you instead assigned some small non-zero number to $\mathrm{P}(\omega_i)$, then we have undesirable conclusions such as

$$\mathrm{P}(\{1, 1/2, 1/3, \ldots\}) = \infty.$$

The way to fix this is to avoid defining the probability of a set in terms of the probability of elementary outcomes and work with more general sets. While we would ideally like to be able to specify the probability of every subset of Ω, it turns out that we cannot do so in a mathematically consistent way. The trick then is to work with a smaller object known as a σ-algebra, that is the set of "nice" subsets of Ω.

Given a sample space Ω, a σ-algebra $\mathcal{F}$ (also called a σ-field) is a collection of subsets of Ω such that

- $\emptyset \in \mathcal{F}$
- If $A \in \mathcal{F}$, then $A^c \in \mathcal{F}$.
- If $A_i \in \mathcal{F}$ for every $i \in \mathbb{N}$, then $\cup_{i=1}^{\infty} A_i \in \mathcal{F}$.

In short, σ-algebra is a collection of subsets of Ω that is closed under complement and countable unions. The pair $(\Omega, \mathcal{F})$, also called a measurable space, is now used to define probability of events. A set A that belongs to $\mathcal{F}$ is called an event. The probability measure

$$\mathrm{P} : \mathcal{F} \to [0, 1].$$

assigns a probability to events in $\mathcal{F}$. We cannot take $\mathcal{F}$ to be too small, e.g., elements of $\mathcal{F} = \{\emptyset, \Omega\}$ are easy to construct our P but are not very useful. For technical reasons, the σ-algebra cannot be too large; notice that we used this concept to *avoid* considering every subset of the sample space $\mathcal{F} = 2^{\Omega}$. Modern probability is defined using a Borel σ-algebra. Roughly speaking, this is an $\mathcal{F}$ that is just large enough to do interesting things but small enough that mathematical technicalities do not occur.

2.1.1 Random variables

A random variable is an assignment of a value to every possible outcome. Mathematically, in our new language of a measurable space, a random variable is a function

$$X : \Omega \to \mathbb{R}$$

if the set $\{\omega : \ X(\omega) \leq c\}$ is $\mathcal{F}$-measurable for every number $c \in \mathbb{R}$. This is equivalent to saying that every preimage of the Borel σ-algebra on reals $\mathcal{B}(\mathbb{R})$ is in $\mathcal{F}$. A statement $X(\omega) = x = 5$ means that the outcome of our experiment happens to $\omega \in \Omega$ when the realized value of the random variable is a particular number x equal to 5 in our case.

ℹ Random variables are typically denoted using capital letters, X, Y, Z although we will be sloppy and not always do so in this course to avoid complicated notation. The distinction between a random variable and the value that it takes will be clear from context.

We can now define functions of random variables, e.g., if X is a random variable, the function $Y = X^3(\omega)$ for every $\omega \in \Omega$, or $Y = X^3$ for short, is a new random variable. An indicator random variable is a special. If $A \subset \Omega$, let $I_A : \Omega \to \{0, 1\}$ be the indicator function of this set A, i.e., $I_A(\omega) = 1$ if ωinA and zero otherwise. If our set $A \in \mathcal{F}$, then I_A is an indicator random variable.

❓ Let us check that Y satisfies our definition of a random variable. If $\{\omega : X(\omega) \leq c\}$ lies in $\mathcal{F}$ then the set $\left\{\omega : Y(\omega) \leq c^{1/3}\right\}$ also lies in $\mathcal{F}$.

ℹ The function I_A is not a random variable if $A \notin \mathcal{F}$, but this is, as we said in the previous section, a mathematical corner case. Most subset of Ω belong to $\mathcal{F}$.

Probability mass functions The probability law, or a probability distribution, of a random variable X is denoted by

$$p_X(x) := \mathrm{P}(X = x) = \mathrm{P}\left(\{\omega \in \Omega : X(\omega) = x\}\right).$$

We denote probability distribution using a lower-case p. It is a function of the realized value x in the range of a random variable, and $p_X(x) \geq 0$ (the probability is non-zero) and $\sum_x p_X(x) = 1$ if X takes on a discrete number of values. For instance, if X is the number of coin tosses until the first head, if we assume that our tosses are independent $\mathrm{P}(H) = p > 0$, then we have

$$p_X(k) = \mathrm{P}(X = k) = \mathrm{P}(TT \cdots TH) = (1-p)^{k-1} p$$

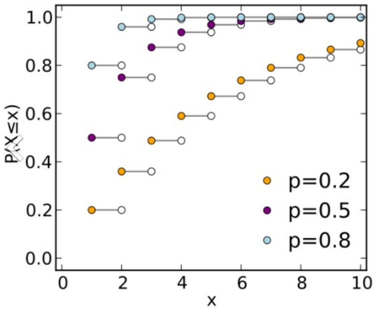
for all $k = 1, 2, \ldots$. This is what is called a geometric probability mass function.

Cummulative distribution function A cummulative distribution function (CDF) is the probability of a random variable X taking a value less than an particular $x \in \mathbb{R}$, i.e.,

$$F_X(x) = \mathrm{P}(X \leq x).$$

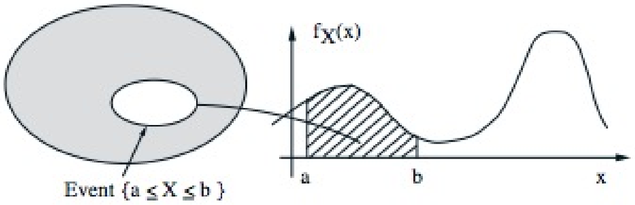
The CDF $F_X(x)$ is a non-decreasing function of x. It converges to zero as $x \to -\infty$ and goes to 1 as $x \to \infty$.

ℹ The CDF of a geometric random variable for different values of p



Note that CDFs need not be continuous, in the case of a geometric r.v. since the values that X takes belong to the set of integers, the CDF is constant between any two integers.

Probability density functions A continuous random variable, i.e., one that takes values in $\mathbb{R}$ is described by a probability density function.



If $F_X(x)$ is the CDF of an r.v. X and X takes values in $\mathbb{R}$, the probability density function (PDF) $f_X(x)$ (or sometimes also denoted by $p_X(x)$) is defined to be

$$\mathrm{P}(a \leq X \leq b) = \int_a^b f_X(x) \, \mathrm{d}x.$$

We also have the following relationship between the CDF and the PDF, the former is the integral of the latter:

$$\mathrm{P}(-\infty \leq X \leq x) = F_X(x) = \int_{-\infty}^{x} f_X(x) \, \mathrm{d}x.$$

This leads to the following interpretation of the probability density function:

$$\mathrm{P}(x \leq X \leq x + \delta) \approx f_X(x) \, \delta.$$

Expectation and Variance The expected value of a random variable X is

$$\mathrm{E}[X] = \sum_x x \, p_X(x)$$

and denotes the center of gravity of the probability mass function. Roughly speaking, it is the average of a large number of repetitions of the same experiment. Expectation is a linear, i.e.,

$$\mathrm{E}[aX + b] = a \, \mathrm{E}[X] + b$$

for any constants a, b. For two independent random variables X, Y we have

$$\mathrm{E}[XY] = \mathrm{E}[X]\,\mathrm{E}[Y].$$

We can also compute the expected value of any function $g(X)$ using the same formula

$$\mathrm{E}[g(X)] = \sum_x g(x)\, p_X(x).$$

In particular, if $g(x) = x^2$ we have the second moment $\mathrm{E}[X^2]$. The variance is defined to be

$$\begin{aligned}\mathrm{var}(X) &= \mathrm{E}\left[(X - \mathrm{E}[X])^2\right] \\ &= \sum_x \left(x - \mathrm{E}[X]\right)^2 p_X(x) \\ &= \mathrm{E}[X^2] - \left(\mathrm{E}[X]\right)^2.\end{aligned}$$

The variance is always non-negative $\mathrm{var}(X) \geq 0$. For an affine function of X, we have

$$\mathrm{var}(aX + b) = a^2\,\mathrm{var}(X).$$

For continuous-valued random variables, the expectation is defined as

$$\mathrm{E}[X] = \int_{-\infty}^{\infty} x p_X(x)\,\mathrm{d}x;$$

the definition of the variance remains the same.

Joint distributions We often wish to think of the joint probability distribution of multiple random variables, say the location of an autonomous car in all three dimensions. The cummulative distribution function associated with this is therefore

$$F_{X,Y,Z}(x, y, z) = \mathrm{P}(X \leq x, Y \leq y, Z \leq z).$$

Just like we have the probability density of a single random variable, we can also write the joint probability density of multiple random variables $f_{X,Y,Z}(x, y, z)$. In this case we have

$$F_{X,Y,Z}(x, y, z) = \int_{-\infty}^{x}\int_{-\infty}^{y}\int_{-\infty}^{z} f_{X,Y,Z}(x, y, z)\mathrm{d}z\mathrm{d}y\mathrm{d}x.$$

The joint probability density factorizes if two random variables are independent:

$$f_{X,Y}(x, y) = f_X(x) f_Y(y) \quad \text{for all } x, y.$$

Two random variables are uncorrelated if and only if

$$\mathrm{E}[XY] = \mathrm{E}[X]\,\mathrm{E}[Y].$$

Note that independence implies uncorrelatedness, they are not equivalent. The covariance is defined as

$$\mathrm{cov}(X, Y) = \mathrm{E}[XY] - \mathrm{E}[X]\,\mathrm{E}[Y].$$

Conditioning As we saw before, for a single random variable X we have

$$\mathrm{P}(x \leq X \leq x+\delta) \approx f_X(x)\,\delta.$$

For two random variables, by analogy we would like

$$\mathrm{P}(x \leq X \leq x+\delta \mid Y \approx y) \approx f_{X|Y}(x \mid y)\delta.$$

The conditional probability density function of X given Y is defined to be

$$f_{X \mid Y}(x \mid y) = \frac{f_{X,Y}(x,y)}{f_Y(y)} \quad \text{if } f_Y(y) > 0.$$

For any given y, the conditional PDF is a normalized section of the joint PDF, as shown below.

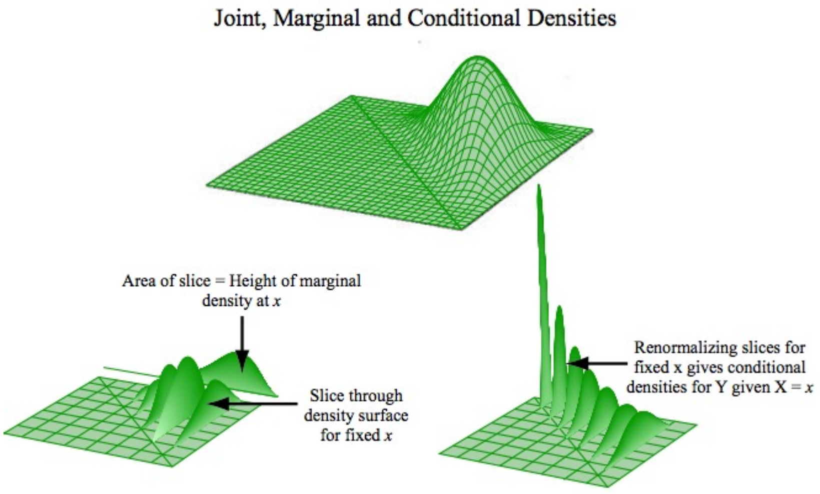

Joint, Marginal and Conditional Densities
Area of slice = Height of marginal density at x
Slice through density surface for fixed x
Renormalizing slices for fixed x gives conditional densities for Y given X = x

Image by MIT OpenCourseWare, adapted from ***Probability***, by J. Pittman, 1999.

Continuous form of Bayes rule We can show using the definition of conditional probability that

$$f_{Y \mid X}(y \mid x) = \frac{f_{X \mid Y}(x \mid y) f_Y(y)}{f_X(x)}. \tag{2.5}$$

Similarly we also have the law of total probability in the continuous form

$$f_X(x) = \int_{-\infty}^{\infty} f_{X \mid Y}(x \mid y)\; f_Y(y)\; \mathrm{d}y.$$

2.2 Using Bayes rule for combining evidence

We now study a prototypical state estimation problem. Let us a consider a robot that is trying to check whether the door to a room is open or not.

We will abstract each observation by the sensors of the robot as a random variable Y. This could be the image from its camera after running some algorithm to check the state of the door, the reading from a laser sensor (if the time-of-flight of the laser is very large then the door is open), or any other mechanism. Let us first distinguish between the cause and effect of the observation: if the door is open, we get an observation of a certain kind (laser sensor says that the time of flight is large), of course the observation is not *caused* by the open door. In other words, we have two kinds of conditional probabilities in this problem

$$\begin{aligned}&\text{P}(\text{open} \mid Y) \text{ is a diagnostic quantity, while}\\ &\text{P}(Y \mid \text{open}) \text{ is a causal quantity.}\end{aligned}$$

Imagine how you would calibrate the sensor in a lab: for each value of the state of the door open, not open you would record all the different observations received Y and calculate the conditional probabilities. The causal probability is much easier to calculate in this context, one may even use some knowledge of elementary physics to model the probability $\text{P}(Y \mid \text{open})$, or one may count the number of times the observation is $Y = y$ for a given state of the door.

The Bayes rule allows us to transform causal knowledge into diagnostic knowledge

$$\text{P}(\text{open} \mid Y) = \frac{\text{P}(Y \mid \text{open})\,\text{P}(\text{open})}{\text{P}(Y)}.$$

Remember that the left hand side (diagnostic) is typically something that we desire to calculate. Let us put some numbers in this formula. Let $\text{P}(Y \mid \text{open}) = 0.6$ and $\text{P}(Y \mid \text{not open}) = 0.3$. We will imagine that the door is open or closed with equal probability: $\text{P}(\text{open}) = \text{P}(\text{not open}) = 0.5$. We then have

$$\begin{aligned}\text{P}(\text{open} \mid Y) &= \frac{\text{P}(Y \mid \text{open})\,\text{P}(\text{open})}{\text{P}(Y)}\\ &= \frac{\text{P}(Y \mid \text{open})\,\text{P}(\text{open})}{\text{P}(Y \mid \text{open})\,\text{P}(\text{open}) + \text{P}(Y \mid \text{not open})\,\text{P}(\text{not open})}\\ &= \frac{0.6 \times 0.5}{0.6 \times 0.5 + 0.3 \times 0.5} = \frac{2}{3}.\end{aligned}$$

Notice something very important, the original (prior) probability of the state of the door is was 0.5. If we have a sensor that fires with higher likelihood if the door is open, i.e., if

$$\frac{\text{P}(Y \mid \text{open})}{\text{P}(Y \mid \text{not open})} > 1$$

then the probability of the door being open after receiving an observation *increases*. If the likelihood were less than 1, then observing a realization of Y would reduce our estimate of the probability of the door being open.

Combining evidence for Markov observations Say we updated the prior probability using our first observation Y_1, let us take another observation Y_2. How can we integrate this new observation? It is again an application of Bayes rule using two observations, or in general multiple observations $Y_1, \ldots, Y_n$. Let us imagine this time that $X =$ open.

ℹ The denominator in Bayes rule, i.e., $\text{P}(Y)$ is called the evidence in statistics.

$$\text{P}(X \mid Y_1, \ldots, Y_n) = \frac{\text{P}(Y_n \mid X, Y_1, \ldots, Y_{n-1})\, \text{P}(X \mid Y_1, \ldots, Y_{n-1})}{\text{P}(Y_n \mid Y_1, \ldots, Y_{n-1})}.$$

Let us make the very natural assumption that says that our observations from the sensor $Y_1, \ldots, Y_n$ are independent given the state of the door X. This is known as the Markov assumption.

We now have

$$\begin{aligned}\text{P}(X \mid Y_1, \ldots, Y_n) &= \frac{\text{P}(Y_n \mid X)\, \text{P}(X \mid Y_1, \ldots, Y_{n-1})}{\text{P}(Y_n \mid Y_1, \ldots, Y_{n-1})} \\ &= \eta\, \text{P}(Y_n \mid X)\, \text{P}(X \mid Y_1, \ldots, Y_{n-1})\end{aligned}$$

where

$$\eta^{-1} = \text{P}(Y_n \mid Y_1, \ldots, Y_{n-1})$$

is the denominator. We can now expand the diagnostic probability on the right-hand side recursively to get

$$\text{P}(X \mid Y_1, \ldots, Y_n) = \prod_{i=1}^{n} \eta_i\, \text{P}(Y_i \mid X)\ \ \text{P}(X). \tag{2.6}$$

where $\eta_i^{-1} = \text{P}(Y_i \mid Y_1, \ldots, Y_{i-1})$.

The calculation in (2.6) is very neat and you should always remember it. Given multiple observations $Y_1, \dots, Y_n$ of the same quantity X, we can compute the conditional probability $\text{P}(X \mid Y_1, \dots, Y_n)$ if we code up two functions to compute

- the causal probability (also called the likelihood of an observation) $\text{P}(Y_i \mid X)$, and
- the denominator η_i^{-1}.

Given these two functions, we can use the recursion to update multiple observations. The same basic idea also holds if you have two quantities to estimate, e.g., $X_1 =$ open door and $X_2 =$ color of the door. The recursive application of Bayes rule lies at the heart of all state estimation methods.

Let us again put some numbers into these formulae, imagine that the observation Y_2 was taken using a different sensor which now has

$$\text{P}(Y_2 \mid \text{open}) = 0.5 \text{ and } \text{P}(Y_2 \mid \text{not open}) = 0.6.$$

We have from our previous calculation that $\text{P}(\text{open} \mid Y_1) = 2/3$ and

$$\begin{aligned}\text{P}(\text{open} \mid Y_1, Y_2) &= \frac{\text{P}(Y_2 \mid \text{open})\,\text{P}(\text{open} \mid Y_1)}{\text{P}(Y_2 \mid \text{open})\,\text{P}(\text{open} \mid Y_1) + \text{P}(Y_2 \mid \text{not open})\,\text{P}(\text{not open} \mid Y_1)} \\ &= \frac{0.5 \times 2/3}{0.5 \times 2/3 + 0.6 \times 1/3} = \frac{5}{8} = 0.625.\end{aligned}$$

Notice in this case that the probability that the door is open has reduced from $\text{P}(\text{open} \mid Y_1) = 2/3$.

2.2.1 Coherence of Bayes rule

Would the probability change if we used sensor Y_2 before using Y_1? In this case, the answer to this question is no and you are encouraged to perform this computation for yourselves. Bayes rule is coherent, it will give the same result regardless of the order of observations.

❓ Can you think of a situation when the order of incorporating observations matters?

The order of incorporating observation matters if the state of the world changes while we make observations, e.g., if we have a sensor that tracks the location of a car, the car presumably moves in between two observations and we would get the wrong answer if our question was “is there a car at this location”.

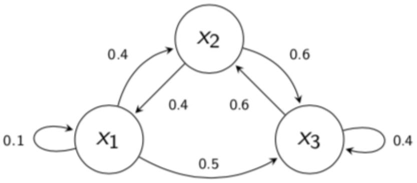
As we motivated in the previous chapter, movement is quite fundamental to robotics and we are typically concerned with estimating the state of a dynamic world around us using our observations. We will next study the concept of a Markov Chain with is a mathematical abstraction for the evolution of the state of the world.

2.3 Markov Chains

Consider the Whack-The-Mole game: a mole has burrowed a network of three holes x_1, x_2, x_3 into the ground. It keeps going in and out of the holes and we are interested in finding which hole it will show up next so that we can give it a nice whack.

- Three holes: $X = \{x_1, x_2, x_3\}$.
- Transition probabilities:

$$T = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.4 & 0 & 0.6 \\ 0 & 0.6 & 0.4 \end{bmatrix}$$



This is an example of a Markov chain. There is a transition matrix T which determines the probability T_{ij} of the mole resurfacing on a given hole x_j given that it resurfaced at hole x^i the last time. The matrix T^k is the k-step transition matrix

$$T_{ij}^k = \mathrm{P}(X_k = x_j \mid X_0 = x_i).$$

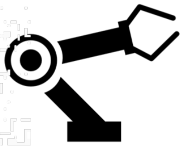
You can see the animations at https://setosa.io/ev/markov-chains to build more intuition.

The key property of a Markov chain is that the next state X_{k+1} is independent of all the past states $X_1, \ldots, X_{k-1}$ given the current state X_k.

$$X_{k+1} \perp\!\!\!\perp X_1, \ldots, X_{k-1} \mid X_k$$

This is known as the Markov property and all systems where we can define a "state" which governs their evolution have this property. Markov chains form a very broad class of systems. For example, all of Newtonian physics fits this assumption.

What is the state of the following systems?



❓ Does a deterministic dynamical system, e.g., a simple pendulum, also satisfy the Markov assumption? What is the transition matrix in this case?

❓ Can you think of a system which does not have the Markov property?

Consider the paramecium above. Its position depends upon a large number of factors: its own motion from the previous time-step but also the viscosity of the material in which it is floating around. One may model the state of the environment around the paramecium as a liquid whose molecules hit thousands of times a second, essentially randomly, and cause disturbances in how the paramecium moves. Let us call this disturbance "noise in the dynamics". If the motion of the molecules of the liquid has some correlations (does it, usually?), this induces correlations in the position of the paramecium. The position of the organism is no longer Markov. This example is important to remember, the Markov property defined above also implies that the noise in the state transition matrix is independent.

Evolution of a Markov chain The probability of being in a state x^i at time $k+1$ can be written as

$$\mathrm{P}(X_{k+1} = x_i) = \sum_{j=1}^{N} \mathrm{P}(X_{k+1} = x_i \mid X_k = x_j)\ \mathrm{P}(X_k = x_j).$$

This equation governs how the probabilities $\mathrm{P}(X_k = x_i)$ change with time k. Let's do the calculations for the Whack-The-Mole example. Say the mole was at hole x_1 at the beginning. So the probability distribution of its presence

$$\pi^{(k)} = \begin{bmatrix} \mathrm{P}(X_k = x_1) \\ \mathrm{P}(X_k = x_2) \\ \mathrm{P}(X_k = x_3) \end{bmatrix}$$

is such that

$$\pi^{(1)} = [1, 0, 0]^\top.$$

We can now write the above formula as

$$\pi^{(k+1)} = T' \pi^{(k)} \tag{2.7}$$

[1] and compute the distribution $\pi^{(t)}$ for all times

$$\begin{aligned}
\pi^{(2)} &= T'\pi^{(1)} = [0.1, 0.4, 0.5]^\top; \\
\pi^{(3)} &= T'\pi^{(2)} = [0.17, 0.34, 0.49]^\top; \\
\pi^{(4)} &= T'\pi^{(3)} = [0.153, 0.362, 0.485]^\top; \\
&\vdots \\
\pi^{(\infty)} &= \lim_{k \to \infty} {T'}^{k}\, \pi^{(1)} \\
&= [0.158, 0.355, 0.487]^\top.
\end{aligned}$$

The numbers $\mathrm{P}(X_k = x_i)$ stop changing with time k. Under certain technical conditions, the distribution $\pi^{(\infty)}$ is unique (single communicating class for a Markov chain finite number states). We can compute this invariant distribution by writing

$$\pi^{(\infty)} = T'\pi^{(\infty)}.$$

We can also compute the distribution $\pi^{(\infty)}$ directly: the invariant distribution is the right-eigenvector of the matrix T' corresponding to the eigenvalue 1.

❓ Do we always know that the transition matrix has an eigenvalue that is 1?

Example 2.1. Consider a Markov chain on two states where the transition

[1] Let us denote the transpose of the matrix T using the Matlab notation T' instead of $T^\top$ for clarity.

matrix is given by

$$T = \begin{bmatrix} 0.5 & 0.5 \\ 0.4 & 0.6 \end{bmatrix}.$$

The invariant distribution is

$$\begin{aligned} \pi^{(1)} &= 0.5\pi^{(1)} + 0.4\pi^{(2)} \\ \pi^{(2)} &= 0.5\pi^{(1)} + 0.6\pi^{(2)}. \end{aligned}$$

Note that the constraint for π being a probability distribution, i.e., $\pi^{(1)}+\pi^{(2)} = 1$ is automatically satisfied by the two equations. We can solve for $\pi^{(1)}, \pi^{(2)}$ to get

$$\pi^{(1)} = 4/9 \quad \pi^{(2)} = 5/9.$$

2.4 Hidden Markov Models (HMMs)

[2]

Markov chains are a good model for how the state of the world evolves with time. We may not always know the exact of these systems and only have sensors, e.g., cameras, LiDARs, and radars, to record observations. These sensors are typically noisy. So we model the observations as random variables.

Hidden Markov Models (HMMs) are an abstraction to reason about observations of the state of a Markov chain. An HMM is a sequence of random variables $Y_1, Y_2, \ldots, Y_n$ such that the distribution of Y_k only depends upon on the hidden state X_k of the associated Markov chain.

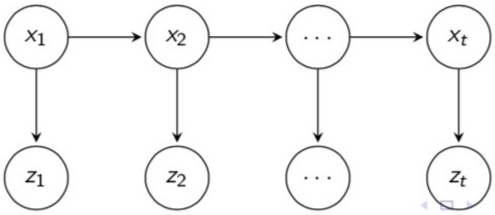


Figure 2.1: A Hidden Markov Model with the underlying Markov chain, the observation at time k only depends upon the hidden state at that time instant. Ignore the notation $Z_1, \ldots, Z_t$ we will denote the observations by Y_k.

Notice that an HMM always has an underlying Markov chain behind it. For example, if we model the position of a car X_k as a Markov chain, our observation of the position at time k would be Y_k. In our example of the robot sensing whether the door is open or closed using multiple observations across time, the Markov chain is trivial, it is simply the transition matrix $\mathrm{P}(\text{not open} \mid \text{not open}) = \mathrm{P}(\text{open} \mid \text{open}) = 1$. Just like Markov chains, HMMs are a very general class of mathematical models that allow us to think about multiple observations across time of a Markov chain.

[2] Parts of this section closely follow Emilio Frazzoli's course notes at https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-410-principles-of-autonomy-and-decision-making-fall-2010/lecture-notes/MIT16_410F10_lec20.pdf and https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-410-principles-of-autonomy-and-decision-making-fall-2010/lecture-notes/MIT16_410F10_lec21.pdf

Let us imagine that the observations of our HMM are also finite in number, e.g., your score in this course $\in [0, 100]$ where the associated state of the Markov chain is your expertise in the subject matter. We will write a matrix of observation probabilities

$$M_{ij} = \mathrm{P}(Y_k = y_j \mid X_k = x_i). \tag{2.8}$$

The matrix M has non-negative entries, after all, each entry is a probability. Since each state has to result in *some* observation, we also have

$$\sum_j M_{ij} = 1.$$

The state transition probabilities of the associated Markov chain are

$$T_{ij} = \mathrm{P}(X_{k+1} = x_j \mid X_k = x_i).$$

Given the abstraction of an HMM, we may be interested in solving a number of problems. We will consider the problem where the state X_k is the position of a car (which could be stationary or moving) and observations Y_k give us some estimate of the this position.

1. **Filtering**: Given observations up to time k, compute the distribution of the state at time k

   $$\mathrm{P}(X_k \mid Y_1, \dots, Y_k).$$

   This is the most natural problem to understand: we want to find the probability of the car being at a location at time k given all previous observations. This is a temporally causal prediction, i.e., we are not using any information from the future to reason about the present.
2. **Smoothing**: Given observations up to time k, compute the distribution of the state at any time $j < k$

   $$\mathrm{P}(X_j \mid Y_1, \dots, Y_k) \quad \text{for } j < k.$$

   The observation at a future time Y_{k+1} gives us some indication of where the car might have been at time k. In this case we are interested in using the entire set of observations from the past $Y_1, \dots, Y_j$, the future $Y_{j+1}, \dots, Y_k$ to estimate the position of the car. Of course, this problem can only be solved *ex post facto*, i.e., after the time instant j. An important thing to remember is that we are interested in the position of the car for all $j < k$ in smoothing.
3. **Prediction**: Given observations up to time k, compute the distribution of the state at a time $j > k$

   $$\mathrm{P}(X_j \mid Y_1, \dots, Y_k) \quad \text{for } j > k.$$

   This is the case when we wish to make predictions about the state of the car $j > k$ given only observations until time k. If we knew the underlying Markov chain for the HMM and its transition matrix T, this would amount to running (2.7) forward using the output of the filtering problem as the initial distribution of the state.

❓ Why is this true?

4. **Decoding**: Find the most likely state trajectory $X_1, \dots, X_k$ that maximizes the probability
$$\mathrm{P}(X_1, \dots, X_k \mid Y_1, \dots, Y_k)$$
given observations $Y_1, \dots, Y_k$. Observe that the smoothing problem is essentially solved independently for all time-steps $j < k$. It stands to reason that if we knew a certain state (say car made a right turn) was likely given observations at time $k+1$ and that the traffic light was green at time k (given our observations of the traffic light), then we know that the car did not stop at the intersection at time k. The decoding problem allows us to reason about the joint probability of the states and outputs the most likely trajectory given all observations.
5. **Likelihood of observations**: Given the observation trajectory, $Y_1, \dots, Y_k$, compute the probability
$$\mathrm{P}(Y_1, \dots, Y_k).$$
As you may recall, this is the denominator that we need for the recursive application of Bayes rule. It is made difficult by the fact that we do not know the state trajectory $X_1, \dots, X_k$ corresponding to these observations.

These problems are closely related with each other and we will next dig deeper into them. We will first discuss two building blocks, called the forward and backward algorithm that together help solve all the above problems.

2.4.1 The forward algorithm

Consider the problem of computing the likelihood of observations. We can certainly write

$$\begin{aligned}
&\mathrm{P}(Y_1, \dots, Y_k) \\
&= \sum_{\text{all } (x_1, \dots, x_k)} \mathrm{P}(Y_1, \dots, Y_k \mid X_1, \dots, X_k)\, \mathrm{P}(X_1, \dots, X_k) \\
&= \sum_{\text{all } (x_1, \dots, x_k)} \prod_{i=1}^{k} \mathrm{P}(Y_i = y_i \mid X_i = x_i)\, \mathrm{P}(X_1 = x_1) \prod_{i=2}^{k} \mathrm{P}(X_k = x_k \mid X_{k-1} = x_{k-1}) \\
&= \sum_{\text{all } (x_1, \dots, x_k)} M_{x_1 y_1}\, M_{x_2 y_2} \dots M_{x_k y_k}\, \pi_{x_1}\, T_{x_1 x_2} \dots T_{x_{k-1} x_k}.
\end{aligned}$$

But this is a very large computation, for each possible trajectory $(x_1, \dots, x_k)$ the states could have taken, we need to perform $2k$ matrix multiplications.

❓ How many possible state trajectories are there? What is the total cost of computing the likelihood of observations?

Forward algorithm We can simplify the above computation using the

Markov property of the HMM as follows. We will define a quantity known as the forward variable

$$\alpha_k(x) = \mathrm{P}(Y_1, \ldots, Y_k, X_k = x) \tag{2.9}$$

where $Y_1, \ldots, Y_k$ is our observation sequence up to time k. Observe now that

1. We can initialize

$$\alpha_1(x) = \pi_x \; M_{x,y_1} \quad \text{for all } x.$$

2. For each time $i = 1, \ldots, k-1$, for all states x, we can compute

$$\alpha_{k+1}(x) = M_{xy_{k+1}} \sum_{x'} \alpha_k(x') \; T_{x'x}.$$

using the law of total probability.

3. Finally, we have

$$\mathrm{P}(Y_1, \ldots, Y_k) = \sum_x \alpha_k(x)$$

by marginalizing over the state variables X_k.

This recursion in the forward algorithm is a powerful idea and is much faster than our naive summation above.

❓ What is the computational complexity of the Forward algorithm?

2.4.2 The backward algorithm

Just like the forward algorithm performs the computation recursively in the forward direction, we can also perform a backward recursion to obtain the probability of the observations. Let us imagine that we have an observation trajectory

$$Y_1, \ldots, Y_t$$

up to some time t. We first define the so-called backward variables which are the probability of a future trajectory given the state of the Markov chain at a particular time instant

$$\beta_k(x) = \mathrm{P}(Y_{k+1}, Y_{k+2}, \ldots, Y_t \mid X_k = x). \tag{2.10}$$

Notice that the backward variables β_k with the conditioning on $X_k = x$ are slightly different than the forward variables α_k which are the joint probability of the observation trajectory and $X_k = x$.

The Backward algorithm We can compute the variables $\beta_k(x)$ recursively again as follows.

1. Initialize
$$\beta_t(x) = 1 \quad \text{for all } x.$$
This simply indicates that since we are the end of the trajectory, the future trajectory $Y_{t+1}, \ldots$ does not exist.
2. For all $k = t-1, t-2, \ldots, 1$, for all x, update
$$\beta_k(x) = \sum_{x'} \beta_{k+1}(x')\ T_{xx'} M_{x' y_{k+1}}.$$
3. We can now compute
$$\mathrm{P}(Y_1, \ldots, Y_t) = \sum_x \beta_1(x)\ \pi_x\ M_{xy_1}.$$

❓ What is the computational complexity of running the backward algorithm?

Implementing the forward and backward algorithms in practice The update equations for both α_k and β_k can be written using a matrix vector multiplication. We maintain the vectors

$$\begin{aligned}\alpha_k &:= [\alpha_k(x_1), \alpha_k(x_2), \ldots, \alpha_k(x_N)]\\ \beta_k &:= [\beta_k(x_1), \beta_k(x_2), \ldots, \beta_k(x_N)]\end{aligned}$$

and can write the updates as

$$\alpha_{k+1}^\top = M_{\cdot, y_{k+1}} \odot \left(\alpha_k^\top\ T\right)$$

where $\odot$ denotes the element-wise product and $M_{\cdot, y_{k+1}}$ is the y_{k+1}^{th} column of the matrix M. The update equation for the backward variables is

$$\beta_k = \left(\beta_{k+1} \odot M_{\cdot, y_{k+1}}\right)\ T.$$

You must be careful about directly implement these recursions however, because we are iteratively multiplying by matrices T, M whose entries are all smaller than 1 (they are all probabilities after all), we can quickly run into difficulties where α_k, β_k become too small for some states and we get numerical underflow. You can implement these algorithms in the log-space by writing similar update equations for $\log \alpha_k$ and $\log \beta_k$ to avoid such numerical issues.

2.4.3 Bayes filter

Let us now use the forward and backward algorithms to solve the filtering problem. We want to compute

$$\mathrm{P}(X_k = x \mid Y_1, \ldots, Y_k)$$

for all states x in the Markov chain. We have that

$$\mathrm{P}(X_k = x \mid Y_1, \dots, Y_k) = \frac{\mathrm{P}(X_k = x, Y_1, \dots, Y_k)}{\mathrm{P}(Y_1, \dots, Y_k)} = \eta\ \alpha_k(x) \qquad (2.11)$$

where since $\mathrm{P}(X_k = x \mid Y_1, \dots, Y_k)$ is a legitimate probability distribution on x, we have

$$\eta = \left(\sum_x \alpha_k(x) \right)^{-1}.$$

As simple as that. In order to estimate the state at time k, we run the forward algorithm to update variables $\alpha_i(x)$ from $i = 1, \dots, k$. We can implement this using the matrix-vector multiplication in the previous section.

This is a commonly used algorithm known as the Bayes filter and is our first insight into state estimation.

An important fact Even if the filtering estimate is computed recursively using each observation as it arrives, the estimate is actually the probability of the current state given *all* past observations.

$$\mathrm{P}(X_k = x \mid Y_1, \dots, Y_k) \neq \mathrm{P}(X_k = x \mid Y_k)$$

This is an extremely important concept to remember, in state-estimation we are always interested in computing the state given all available observations. In the same context, is the following statement true?

$$\mathrm{P}(X_k = x \mid Y_1, \dots, Y_k) = \mathrm{P}(X_k = x \mid Y_k, X_{k-1})$$

2.4.4 Smoothing

For smoothing given observations till time t, we would like to compute

$$\mathrm{P}(X_k = x \mid Y_1, \dots, Y_t)$$

for all time instants $k = 1, \dots, t$. Observe the filtering

$$\begin{aligned}
\mathrm{P}(X_k = x \mid Y_1, \dots, Y_t) &= \frac{\mathrm{P}(X_k = x, Y_1, \dots, Y_t)}{\mathrm{P}(Y_1, \dots, Y_t)} \\
&= \frac{\mathrm{P}(X_k = x, Y_1, \dots, Y_k, Y_{k+1}, \dots, Y_t)}{\mathrm{P}(Y_1, \dots, Y_t)} \\
&= \frac{\mathrm{P}(Y_{k+1}, \dots, Y_t \mid X_k = x, Y_1, \dots, Y_k)\ \mathrm{P}(X_k = x, Y_1, \dots, Y_k)}{\mathrm{P}(Y_1, \dots, Y_t)} \\
&= \frac{\mathrm{P}(Y_{k+1}, \dots, Y_t \mid X_k = x)\ \mathrm{P}(X_k = x, Y_1, \dots, Y_k)}{\mathrm{P}(Y_1, \dots, Y_t)} \\
&= \frac{\beta_k(x)\ \alpha_k(x)}{\mathrm{P}(Y_1, \dots, Y_t)}
\end{aligned} \qquad (2.12)$$

Study the first step carefully, the numerator is *not* equal to $\alpha_k(x)$ because observations go all the way till time t. The final step uses both the Markov and the HMM properties: future observations $Y_{k+1}, \dots, Y_t$ depend only upon future states $X_{k+1}, \dots, X_t$ (HMM property) which are independent of the past observations and states give the current state $X_k = x$ (Markov property).

Smoothing can therefore be implemented by running the forward algorithm to update α_k from $k = 1, \dots, t$ and the backward algorithm to update β_k from time $k = t, \dots, 1$.

To see an example of smoothing in action, see ORB-SLAM 2. What do you think is the state of the Markov chain in this video?

❓ Both the filtering problem and the smoothing problem give us the probability of the state given observations. Discuss which one should we should use in practice and why?

Example for the Whack-the-mole problem Let us assume that we do not see which hole the mole surfaces from (say it is dark outside) but we can hear it. Our hearing is not very precise so we have an observation probabilities

$$M = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}.$$

Assume that the mole surfaces three times and we make the measurements

$$Y_1 = 1, Y_2 = 3, Y_3 = 3.$$

We want to compute the distribution of the states the mole could be in at each time. Assume that the we know the mole was in hole 1 at the first step, i.e., $\pi_1 = (1, 0, 0)$ for the Markov chain, like we had in Section 2.3.

Run the forward backward algorithm and see that

$$\alpha_1 = (0.6, 0, 0)\,, \alpha_2 = (0.012, 0.048, 0.18)\,, \alpha_3 = (0.0041, 0.0226, 0.0641)\,,$$

and

$$\beta_3 = (1, 1, 1)\,, \beta_2 = (0.4, 0.44, 0.36)\,, \beta_1 = (0.1512, 0.1616, 0.1392)\,.$$

Using these, we can now compute the filtering and the smoothing state distributions, let us denote them by π^f and π^s respectively.

$$\pi_1^f = (1, 0, 0)\,, \pi_2^f = (0.05, 0.2, 0.75), \pi_3^f = (0.045, 0.2487, 0.7063)$$

and

$$\pi_1^s = (1, 0, 0)\,, \pi_2^s = (0.0529, 0.2328, 0.7143), \pi_3^s = (0.045, 0.2487, 0.7063).$$

❓ Do you notice any pattern in the solution returned by the filtering and the smoothing problem? Explain why that is the case.

2.4.5 Prediction

We would like to compute the future probability of the state give observations up to some time

$$\mathrm{P}(X_k = x \mid Y_1, \dots, Y_t) \quad \text{for } t < k.$$

Here is a typical scenario when you would need this estimate. Imagine that you are tracking the position of a car using images from your camera. You are using a deep network to detect the car in each image Y_k and since the neural network is quite slow, the car moves multiple time steps forward before you get the next observation. As you can appreciate, it would help us compute a more accurate estimate of the conditional probability of $X_k = x$ if we propagated the position of the car in between successive observations using our Markov chain. This is easy to do.

1. We compute the filtering estimate $\pi_t^f = \mathrm{P}(X_t = x \mid Y_1, \ldots, Y_t)$, using the forward algorithm.
2. Propagate the Markov chain forward for $k - t$ time-steps using π_t^f as the initial condition using

$$\pi_{i+1} = T' \pi_i.$$

2.4.6 Decoding: Viterbi's Algorithm

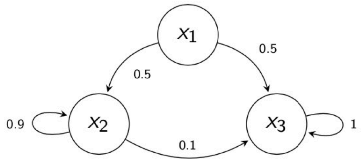
Both filtering and smoothing calculate the probability distribution of the state at time k. For instance, after recording a few observations, we can compute the probability distribution of the position of the car at each time instant. How do we get most likely trajectory of the car? One option is to choose

$$\hat{X}_k = \underset{x}{\text{argmax}}\ \mathrm{P}(X_k = x \mid Y_1, \ldots, Y_t)$$

at each instant and output

$$(\hat{X}_1, \ldots, \hat{X}_t)$$

as the answer. This is however only the point-wise best estimate of the state. This sequence may not be the most likely trajectory of Markov chain underlying our HMM. In the decoding problem, we are interested in computing the most likely state trajectory, not the point-wise most likely sequence of states. Let us take an example of the Whack-the-mole again. We will use a slightly different Markov chain shown below.



$$T = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0 & 0.9 & 0.1 \\ 0 & 0.1 & 0.9 \end{bmatrix} \qquad M = \begin{bmatrix} 0.5 & 0.5 \\ 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$$

There are three states x_1, x_2, x_3 with known initial distribution $\pi = (1, 0, 0)$ and transition probabilities and observations given by matrices T, M respectively. Let us say that we only have two observations $\{y_2, y_3\}$ this time and get the observation sequence

$$(2, 3, 3, 2, 2, 2, 3, 2, 3)$$

from our sensor. The filtering estimates are as follows.

t	x_1	x_2	x_3
1	**1.0000**	0	0
2	0	0.1000	**0.9000**
3	0	0.0109	**0.9891**
4	0	0.0817	**0.9183**
5	0	0.4165	**0.5835**
6	0	**0.8437**	0.1563
7	0	0.2595	**0.7405**
8	0	**0.7328**	0.2672
9	0	0.1771	**0.8229**

The most likely state at each instant is marked in blue. The point-wise most likely sequence of states is

$$(1, 3, 3, 3, 3, 2, 3, 2, 3).$$

Observe that this is not even feasible for the Markov chain. The transition from $x_3 \to x_2$ is not even possible so this answer is clearly wrong. Let us look at the smoothing estimates.

t	x_1	x_2	x_3
1	**1.0000**	0	0
2	0	**0.6297**	0.3703
3	0	**0.6255**	0.3745
4	0	**0.6251**	0.3749
5	0	**0.6218**	0.3782
6	0	**0.5948**	0.4052
7	0	0.3761	**0.6239**
8	0	0.3543	**0.6457**
9	0	0.1771	**0.8229**

The point-wise most likely states in this case are feasible

$$(1, 2, 2, 2, 2, 2, 3, 3, 3).$$

Because the smoothing estimate at time k also takes into account the observations from the future $t > k$, it effectively eliminates the impossible transition $x_3 \to x_2$. This is still not however the most likely trajectory.

We will exploit the Markov property again to calculate the most likely state trajectory recursively. Let us define the "decoding variables" as

$$\delta_k(x) = \max_{(X_1,\ldots,X_{k-1})} \mathrm{P}(X_1 = X_1, \ldots, X_{k-1} = X_{k-1}, X_k = x, Y_1, \ldots, Y_k); \tag{2.13}$$

this is the joint probability of the most likely state trajectory that ends at the state x at time k while generating observations $Y_1, \ldots, Y_k$. We can now see that

$$\delta_{k+1}(x) = \max_{x'} \; \delta_k(x') \, T_{x'x} \, M_{x,y_{k+1}}; \tag{2.14}$$

the joint probability that the most likely trajectory ends up at state x at time $k+1$ is the maximum of among the joint probabilities that end up at any state

x' at time k multiplied by the one-step state transition $T_{x'x}$ and observation $M_{xy_{k+1}}$ probabilities. We would like to iterate upon this identity to find the most likely path. The key idea is to maintain a pointer to the parent state $\text{parent}_k(x)$ of the most likely trajectory, i.e., the state from which you could have reached $X_k = x$ given observations. Let us see how.

Viterbi's algorithm First initialize

$$\begin{aligned}\delta_1(x) &= \pi_x \; M_{xy_1}\\ \text{parent}_k(x) &= \text{null}.\end{aligned}$$

for all states x. For all times $k = 1, \ldots, t-1$, for all states x, update

$$\begin{aligned}\delta_{k+1}(x) &= \max_{x'} \; \delta_k(x') \; T_{x'x} \; M_{x,y_{k+1}}\\ \text{parent}_{k+1}(x) &= \underset{x'}{\text{argmax}} \; \left(\delta_k(x') \; T_{x'x}\right).\end{aligned}$$

The most likely final state is

$$\hat{x}_t = \underset{x'}{\text{argmax}} \; \delta_t(x')$$

and we can now backtrack using our parent pointers to find the most likely trajectory that leads to this state

$$\hat{x}_k = \text{parent}_{k+1}(\hat{x}_{k+1}).$$

The most likely trajectory given observations is

$$\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_t$$

and the joint probability of this trajectory and all observations is

$$\text{P}(X_1 = \hat{x}_1, \ldots, X_t = \hat{x}_t, Y_1 = y_1, \ldots, Y_t = y_t) = \delta_t(\hat{x}_t).$$

This is a very widely used algorithm, both in robotics and in other areas such as speech recognition (given audio, find the most likely sentence spoken by the person), wireless transmission and reception, DNA analysis (e.g., the state of the Markov chain is the sequence ACTG... and our observations are functions of these states at periodic intervals). Its name comes from Andrew Viterbi who developed the algorithm in the late 60s, he is one of the founders of Qualcomm Inc.

Here is how Viterbi's algorithm would look like for our whack-the-model example.

$$\begin{aligned}\delta_1 = (0.6, 0, 0), \delta_2 = (0.012, 0.048, 0.18), \delta_3 = (0.0038, 0.0216, 0.0432)\\ \text{parent}_1 = (\text{null}, \text{null}, \text{null}), \text{parent}_2 = (1, 1, 1), \text{parent}_3 = (2, 3, 3).\end{aligned}$$

The most likely path is the one that ends in 3 with joint probability 0.0432.

This path is (1, 3, 3).

Let us also compute Viterbi's algorithm for a longer observation sequence.

t	x_1	x_2	x_3
1	0.5/0	0	0
2	0/1	0.025/1	0.225/1
3	0/1	0.00225/2	0.2025/3
4	0/1	0.0018225/2	0.02025/3
5	0/1	0.0014762/2	0.002025/3
6	0/1	0.0011957/2	0.0002025/3
7	0/1	0.00010762/2	0.00018225/3
8	0/1	8.717e-05/2	1.8225e-05/3
9	0/1	7.8453e-06/2	1.6403e-05/3

The most likely trajectory is

$$(1, 3, 3, 3, 3, 3, 3, 3, 3).$$

Notice that if we had only 8 observations, the most likely trajectory would be

$$(1, 2, 2, 2, 2, 2, 2, 2, 2).$$

Just like the Bayes filter, Viterbi's algorithm is typically implemented using $\log \delta_k(x)$ to avoid numerical underflows. This is particularly important for Viterbi's algorithm: since $\delta_k(x)$ is the probability of an entire state and observation trajectory it can get small very quickly for unlikely states (as seen in the figure).

What is the computational complexity of Viterbi's algorithm? It is line in the time-horizon t and quadratic in the number of states in the Markov chain. We are plucking out the most likely trajectory out of $\text{card}(X)^t$ possible trajectories using the δ_k variables. Does this remind you of some other problem that you may have seen before?

2.4.7 Shortest path on a Trellis graph

You may have seen Dijkstra's algorithm before that computes the shortest path to reach a node in the graph given costs of traversing every edge.

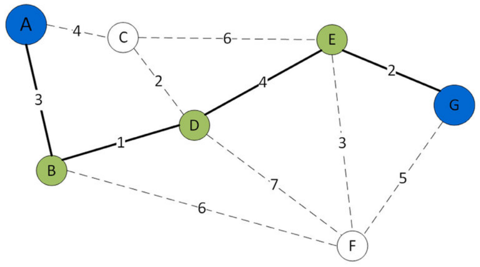


Figure 2.2: A graph with costs assigned to every edge. Dijkstra's algorithm finds the shortest path in this graph between nodes A and B using dynamic programming.

In the case of Viterbi's algorithm, we are also interested in finding the most likely path. For example we can write our joint probabilities as

$$\mathrm{P}(X_1, X_2, X_3 \mid Y_1, Y_2, Y_3) = \frac{\mathrm{P}(Y_1 \mid X_1)\ \mathrm{P}(Y_2 \mid X_2)\ \mathrm{P}(Y_3 \mid X_3)\ \mathrm{P}(X_1)\,\mathrm{P}(X_2 \mid X_1)\,\mathrm{P}(X_3 \mid X_2)}{\mathrm{P}(Y_1, Y_2, Y_3)}.$$

$$\begin{aligned}\Rightarrow \log \mathrm{P}(X_1, X_2, X_3 \mid Y_1, Y_2, Y_3) &= \log \mathrm{P}(Y_1 \mid X_1) + \log \mathrm{P}(Y_2 \mid X_2) + \log \mathrm{P}(Y_3 \mid X_3) \\ &\quad + \log \mathrm{P}(X_1) + \log \mathrm{P}(X_2 \mid X_1) + \log \mathrm{P}(X_3 \mid X_2) - \log \mathrm{P}(Y_1, Y_2, Y_3).\end{aligned}$$

To find the most likely trajectory, we want to minimize $-\log \mathrm{P}(X_1, X_2, X_3 \mid Y_1, Y_2, Y_3)$. The term $\log \mathrm{P}(Y_1, Y_2, Y_3)$ does not depend on X_1, X_2, X_3 and is a constant as far as the most likely path given observations is concerned. We can now write down the "Trellis" graph as shown below.

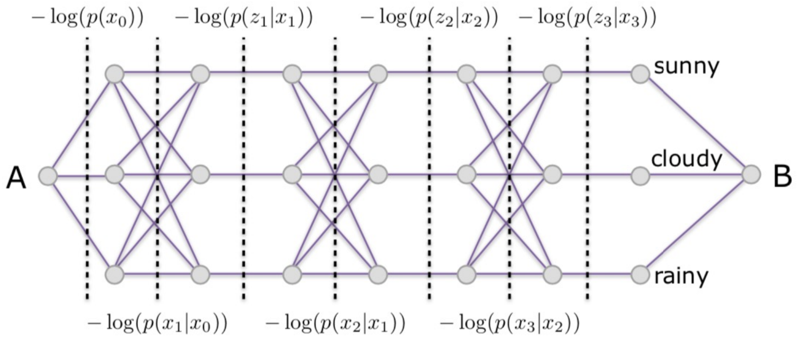


Figure 2.3: A Trellis graph for a 3-state HMM for a sequence of three observations. Disregard the subscript x_0.

Each edge is either the log-probability of the transition of the Markov chain, or it is the log-probability of the receiving the observation given a state. We create a dummy initial node A and a dummy terminal node B. The edge-costs of the final three states, in this case sunny/cloudy/rainy, are zero. The costs from node A to the respective states are the log-probabilities of the initial state distribution. Dijkstra's algorithm, which we will study in Module 2 in more details, now gives the shortest path on the Trellis graph. This approach is the same as that of the Viterbi's algorithm: our parent pointers $\text{parent}_k(x)$ are the parent nodes in Dijkstra's algorithm and our delta variables $\delta_k(x)$ is the cost of each node in the Trellis graph maintained by the Dijkstra's algorithm.

2.5 Learning an HMM from observations

In the previous sections, given an HMM that had an initial distribution π for the Markov chain, a transition matrix T for the Markov chain and an observation matrix M

$$\lambda = (\pi, T, M)$$

we computed various quantities such as

$$\mathrm{P}(Y_1, \ldots, Y_t;\ \lambda)$$

for an observation sequence $Y_1, \dots, Y_t$ of the HMM. Given an observation sequence, we can also go back and update our HMM to make this observation sequence more likely. This is the simplest instance of *learning* an HMM. The prototypical problem to imagine that our original HMM λ comes from our knowledge of the original problem (say a physics model of the dynamics of a robot and its sensors). Given more data, namely the observations, we want to update this model. The most natural way to update the model is to maximize the likelihood of observations given our model, i.e.,

$$\lambda^* = \underset{\lambda}{\text{argmax}}\, \mathrm{P}(Y_1, \dots, Y_t;\ \lambda).$$

This is known as maximum-likelihood estimation (MLE). In this section we will look at the Baum-Welch algorithm which solves the MLE problem iteratively. Given λ, it finds a new HMM $\lambda' = (\pi', T', M')$ (the $'$ denotes a new matrix, not the transpose here) such that

$$\mathrm{P}(Y_1, \dots, Y_t;\ \lambda') > \mathrm{P}(Y_1, \dots, Y_t;\ \lambda).$$

Let us consider a simple problem. We are going to imagine that the FBI is trying to catch the dangerous criminal Keyser Soze who is known to travel between two cities Los Angeles (LA) which will be state x_1 and New York City (NY) which will be state x_2. The FBI initially have no clue about his whereabouts, so their initial belief on his location is uniform $\pi = [0.5, 0.5]$. His movements are modeled using a Markov chain

$$T = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix},$$

e.g., if Soze is in LA, he is likely to stay in LA or go to NY with equal probability. The FBI can make observations about him, they either observe him to be in LA (y_1), NY (y_2) or do not observe anything at all (null, y_3).

$$M = \begin{bmatrix} 0.4 & 0.1 & 0.5 \\ 0.1 & 0.5 & 0.4 \end{bmatrix}.$$

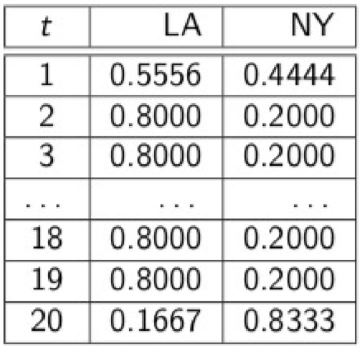
Say that they received an observation sequence of 20 periods

(null, LA, LA, null, NY, null, NY, NY, NY, null, NY, NY, NY, NY, NY, null, null, LA, LA, NY).

Can we say something about the probability of Soze's movements? At each time k we can compute

$$\gamma_k(x) := \mathrm{P}(X_k = x \mid Y_1, \dots, Y_t)$$

the smoothing probability. We can also compute the most likely state trajectory he could have take given our observations using decoding. Let us focus on the smoothing probabilities $\gamma_k(x)$ as shown below.



t	LA	NY
1	0.5556	0.4444
2	0.8000	0.2000
3	0.8000	0.2000
...	...	...
18	0.8000	0.2000
19	0.8000	0.2000
20	0.1667	0.8333

The point-wise most likely sequence of states after doing so turns out to be

$$(\text{LA, LA, LA, LA, NY, LA, NY, NY, NY, LA, NY, NY, NY, NY, NY, LA, LA, LA, LA, NY}).$$

Notice how smoothing fills in the missing observations above.

Expected state visitation counts The next question we should ask is how should we update the model λ given this data. We are going to learn the entries of the state-transition using

$$T'_{x,x'} = \frac{\text{E[number of transitions from } x \text{ to } x']}{\text{E[number of times the Markov chain was in state } x]}.$$

What is the denominator, it is simply the sum of the probabilities that the Markov chain was at state x at time $1, 2, \dots, t-1$ given our observations, i.e.,

$$\text{E[number of times the Markov chain was in state } x] = \sum_{k=1}^{t-1} \gamma_k(x).$$

The numerator is given in a similar fashion. We will define a quantity

$$\begin{aligned} \xi_k(x, x') &:= \text{P}(X_k = x, X_{k+1} = x' \mid Y_1, \dots, Y_t) \\ &= \eta\ \alpha_k(x) T_{x,x'} M_{x', y_{k+1}}\ \beta_{k+1}(x'); \end{aligned} \tag{2.15}$$

❓ Derive the expression for $\xi_k(x, x')$ for yourself.

where η is a normalizing constant such that $\sum_{x,x'} \xi_k(x, x') = 1$. Observe that ξ_k is the joint probability of X_k and X_{k+1}

$$\begin{aligned} \xi_k(x, x') &= \text{P}(X_{k+1} = x' \mid X_k = x, Y_1, \dots, Y_t)\ \gamma_k(x) \\ &\neq T_{x,x'}\ \gamma_k(x) \\ &= \text{P}(X_{k+1} = x' \mid X_k = x)\ \text{P}(X_k = x \mid Y_1, \dots, Y_t). \end{aligned}$$

The expected value of transitioning between states x and x' is

$$\text{E[number of transitions from } x \text{ to } x'] = \sum_{k=1}^{t-1} \xi_k(x, x').$$

This gives us our new state transition matrix, you will see in the homework that it comes to be

$$T' = \begin{bmatrix} 0.47023 & 0.52976 \\ 0.35260 & 0.64739 \end{bmatrix}.$$

This is a much better informed FBI than the other we had before beginning the problem where the transition matrix was all 0.5s.

The new initial distribution What is the new initial distribution for the HMM? Recall that we are trying to compute the best HMM given the observations, so if the initial distribution was

$$\pi = \mathrm{P}(X_1)$$

before receiving any observations from the HMM, it is now

$$\pi' = \mathrm{P}(X_1 \mid Y_1, \dots, Y_t) = \gamma_1(x);$$

the smoothing estimate at the first time-step.

Updating the observation matrix We can use a similar logic at the expected state visitation counts to write

$$\begin{aligned} M'_{x,y} &= \frac{\mathrm{E}[\text{number of times in state } x, \text{ when observation was } y]}{\mathrm{E}[\text{number of times the Markov chain was in state } x]} \\ &= \frac{\sum_{k=1}^{t} \gamma_k(x) \mathbf{1}_{\{y_k = y\}}}{\sum_{k=1}^{t} \gamma_k(x)}. \end{aligned}$$

You will see in your homework problem that this matrix comes up to be

$$M' = \begin{bmatrix} 0.39024 & 0.20325 & 0.40650 \\ 0.06779 & 0.706214 & 0.2259 \end{bmatrix}.$$

Notice how the observation probabilities for the unknown state y_3 have gone down because the Markov chain does not have those states.

The ability to start with a rudimentary model of the HMM and update it using observations is quite revolutionary. Baum et al. proved in the paper Baum, Leonard E., et al. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains." The annals of mathematical statistics 41.1 (1970): 164-171. Discuss the following questions:

- When do we stop in our iterated application of the Baum-Welch algorithm?
- Are we always guaranteed to find the same HMM irrespective of our initial HMM?
- If our initial HMM λ is the same, are guaranteed to find the same HMM λ' across two different iterations of the Baum-Welch algorithm?
- How many observations should we use to update the HMM?