

COMP3667: Reinforcement learning practical 1

Notation, glossary, concepts, and setup

robert.lieck@durham.ac.uk

This is the version *with* answers!

Contents

1	Overview	1
2	Probability distributions	1
2.1	Notation	2
2.2	Probability distributions over one variable	2
2.2.1	Probability <i>density</i> versus probability <i>mass</i>	4
2.3	Probability distributions over multiple variables	4
2.3.1	Joint probability distributions	4
2.3.2	Marginal and conditional probability distributions	6
3	Independence	10
3.1	Graphical models	11
4	Expectations	12

1 Overview

Note 1: *There is a lot of maths covered in this practical and it is completely natural for some to feel rusty or overwhelmed by it. For those of you, don't panic! The math covered here is certainly useful, but mastery of it is not required to design, train and test your RL agents. You will still be able to successfully complete the course and have fun with the coding, which is the much bigger part!*

Note 2: *There is a lot of material in this practical. You might not be able to complete all the questions/exercises before the practical; try to sample uniformly from the different sections. We will go through everything in the practical and answer any questions you might have. You also have a version with answers, but you should definitely try to solve everything by yourself first.*

Welcome to the first reinforcement learning practical. This practical assumes you have already completed the first deep learning practical, and therefore have an appropriate training environment setup. If you have not yet done this, please complete that first. This practical has the following aims:

- Review some statistics & concepts used in the module
- Introduce a few foundational reinforcement learning concepts
- Chat with the staff about any questions about the module

2 Probability distributions

To understand reinforcement learning, we have to think probabilistically. First, the world is not a deterministic place, at least not from an RL agent's point of view. Second, it sometimes is important to have some variation/stochasticity in the agent's actions and not always do the same thing in the same situation over and over again. Therefore, we will revisit a couple of notations, definitions, and principles to deal with probability distributions that are relevant for this lecture. Probability distributions occur, for instance, as

- the agent’s policy, $\pi(a | s)$, which is a probability distribution over actions a conditional on the environment’s state s .
- the environment’s transition function, $p(s' | s, a)$, which is a probability distribution over the next state s' conditional on the previous state s and the agent’s action a .

Why would you use a stochastic policy in RL?

- Recall from the first lecture, we often wish our agent to learn a policy that can be **stochastic**, where:

$$a \sim \pi(a|s).$$

- In other words, rather than always taking the same action for a given state as in a deterministic policy $a = \pi(s)$, we learn the distribution of actions for a given state and sample from it to choose our action.
- For continuous actions, like deciding how much force to apply to the joints/motors/actuators in a robot, we most often restrict ourselves to a multivariate normal distribution (multiple dimensions for each joint/motor/actuator) and try to learn the means μ and covariances Σ of these forces before sampling from $\mathcal{N}(a; \mu, \Sigma)$.
- This is important, as it means the robot will not always behave predictably and get stuck in local minima in the environment—it will still take some random actions, but in a sensible way.
- Alternatively, the agent may have a discrete action space—for example it could choose to press a set of buttons on a game controller or not—a probability mass function.

2.1 Notation

$P(a)$ or $P(A)$	A probability distribution over a discrete variable, i.e., a probability <i>mass</i> function (PMF)
$p(a)$ or $p(A)$	A probability distribution over a continuous variable, i.e., a probability <i>density</i> function (PDF), or over a variable whose type has not been specified
$p(a, b)$	A joint probability distribution over two variables
$p(a b)$	A conditional probability distribution of a conditional on b
$p(a = 5)$	The probability (mass or density) of variable a taking the value 5
$a \sim P$	Random variable a has distribution P
$\mathbb{E}_{x \sim P}[f(x)]$ or $\mathbb{E}f(x)$ or $\mathbb{E}_{P(x)}f(x)$	Expectation of $f(x)$ with respect to $P(x)$
$\text{Var}(f(x))$	Variance of $f(x)$ under $P(x)$
$\text{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(x)$
$\mathcal{N}(x; \mu, \Sigma)$	Gaussian/normal distribution over x with mean μ and covariance Σ (x and μ are vectors; Σ is a positive definite matrix)

2.2 Probability distributions over one variable

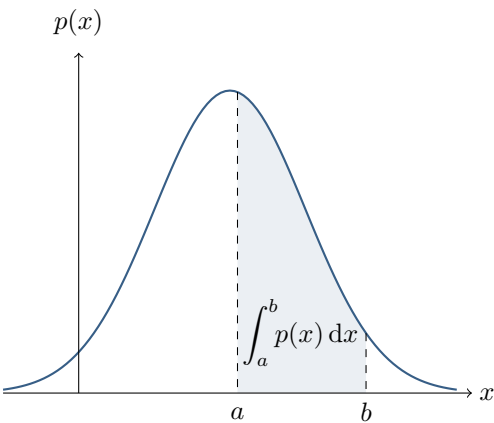
A probability distribution p over a random variable X has to fulfil two conditions:

	Full Notation	Short Notation
Positive Definiteness:	$\forall x \in \text{dom}(X) : p(X = x) \stackrel{!}{\geq} 0$	$p(X) \stackrel{!}{\geq} 0$ (1)
Normalisation (continuous):	$\int_{\text{dom}(X)} p(X = x) \, dx \stackrel{!}{=} 1$	$\int_X p(X) \, dX \stackrel{!}{=} 1$ (2)
(discrete):	$\sum_{x \in \text{dom}(X)} P(X = x) \stackrel{!}{=} 1$	$\sum_X P(X) \stackrel{!}{=} 1$. (3)

Here, $\text{dom}(X)$ means the *domain* of X , that is, all values the variable can take. This might be \mathbb{N} or $\{\text{“A”}, \text{“B”}, \text{“C”}\}$ (for a discrete variable) or \mathbb{R}^n (for a continuous vectorised variable), or something more complicated (e.g. a mixed discrete-continuous domain). Usually, this is not problematic and clear from context, so that we can leave it out as done in the short notation. Similarly, it is clear that in the short notation the variable needs to be assigned to a specific value to compute the integral etc. So, while being mathematically incorrect, we can use the sloppy short notation when it is obvious how to “fix” it. (Please, don’t invent your own short notations that only you know how to fix!)

Example

Let's say that the mean height of people in Durham is $\mu = 178\text{cm}$ with standard deviation $\sigma = 8\text{cm}$. We could say that a random person in Durham has height x with a probability modelled by the normal distribution:

$$p(x | \mu, \sigma^2) = \underbrace{\frac{1}{\sqrt{2\pi\sigma^2}}}_{\text{normalise so } \int p(x)dx = 1} e^{-\frac{1}{2} \underbrace{\left(\frac{x-\mu}{\sigma}\right)^2}_{\substack{\text{shift by mean} \\ \text{scale by STD}}}} \quad (4)$$


Questions

- Graph e^{-x^2} on <https://www.desmos.com/calculator>.
 - Evaluate $\int_{-\infty}^{\infty} e^{-x^2} dx$ in desmos to see it does not integrate to 1 without renormalising.
 - Fix the normalisation and add $\sigma = 8\text{cm}$ and $\mu = 178\text{cm}$.
 - Compute the probability of a person being less than 170cm tall.
- Sample from a multivariate normal distribution using PyTorch.

Listing 1: Sampling from a multivariate normal distribution

```
1 # a 3D multivariate normal distribution with means [1, 7, 3] and identity covariance
2 m = torch.distributions.MultivariateNormal(torch.tensor([1.0, 7.0, 3.0]), torch.eye(3))
3 m.sample()
4
5 # here's a couple of samples around these means (mu's)
6 tensor([0.9872, 7.1267, 1.9909])
7 tensor([1.6446, 8.7120, 3.3128])
```

- What is the domain of a normally distributed random variable? What problems do you anticipate when modelling the height of people with a normal distribution?

The domain are all real numbers \mathbb{R} from $-\infty$ to ∞ . This includes negative numbers, which is a problem if we want to model a strictly positive quantity such as height.

- The value of a normal distribution at its maximum may be larger than one. How can this be? Shouldn't it describe probabilities, which cannot be larger than one?

The value of a PDF is a *density*, not a probability. To get probabilities from a PDF, we need to integrate over a subset of its domain. This effectively scales everything by the volume/area of the subset we integrate over. So, as long as that area is smaller than one, the PDF itself can be larger than one. The requirement of normalisation (i.e. that the integral over the entire domain equal one) ensures that integrals over proper subsets are always less than one and thus correspond to proper probabilities.

- What is the probability of a person being exactly 178cm tall?

The probability of a person being exactly 178cm tall is zero. The value 178cm is just a point in the domain of the variable, which has zero volume. If we would want to know the probability of a person being “approximately 178cm tall”, we would need to define what “approximately” means. This could e.g. be done by using an ϵ environment around 178cm, which we can then integrate over to get a (non-zero) probability.

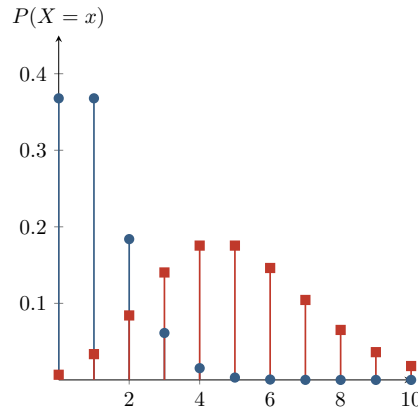
- What do your samples look like if you initialise the covariance as `0.000001*torch.eye(3)` instead?

They are all very close to `[1.0, 7.0, 3.0]` because the variance is so small that samples are effectively (up to machine precision) deterministically reproducing the mean.

2.2.1 Probability *density* versus probability *mass*

Above, we had an example of a probability *density* function: the normal distribution for the continuous variable describing a person's height. Probability *mass* functions are for discrete variables and they work essentially the same, except that integrals are replaced with sums, as in equation 3. To make it explicit that we are dealing with a discrete variable, we may use a capital P instead of the generic lower-case p .

Example



- This figure shows two probability mass functions corresponding to a Poisson distribution with $\lambda = 1$ and $\lambda = 5$, respectively,

$$\text{Pois}(\lambda) := \frac{\lambda^k e^{-\lambda}}{k!} . \quad (5)$$

- If you sum up the values, you will get 1.

Questions

- Sample from a probability mass function (a discrete categorical distribution) with specified probabilities for each index (that must sum to 1) using either
 - `numpy.random.choice(...)` or
 - `torch.distributions.Categorical(...)`.

Listing 2: Sampling from a categorical distribution

```
1 # NumPy version
2 np.random.choice(5, 20, p=[0.1, 0.1, 0.1, 0.6, 0.1])
3 -> array([3, 3, 3, 2, 3, 3, 3, 1, 0, 3, 2, 3, 3, 4, 3, 3, 3, 3, 1])
4
5 # PyTorch version
6 c = torch.distributions.Categorical(torch.tensor([0.1, 0.1, 0.1, 0.6, 0.1]).repeat(20,1))
7 c.sample()
8 -> tensor([2, 3, 0, 3, 3, 3, 4, 1, 1, 4, 3, 0, 4, 4, 3, 3, 0, 0, 3, 3])
```

- The value of a probability *density* function may be greater than one. Is this also true for a probability *mass* function?

No! The value of a PMF *is* a probability (the probability of the variable taking that specific value), therefore, it cannot be greater than one.

2.3 Probability distributions over multiple variables

2.3.1 Joint probability distributions

A random variable seldom comes alone. Let us assume we have two random variables X and Y (the generalisation to an arbitrary number of variables is quite intuitive). Now, an *event* is characterised by both of these variables taking a specific value, which can equivalently be thought of as a single variable

$$Z := (X, Y) \quad \text{with} \quad \text{dom}(Z) := \text{dom}(X) \times \text{dom}(Y) \quad (6)$$

taking a value from the product domain $\text{dom}(X) \times \text{dom}(Y)$.

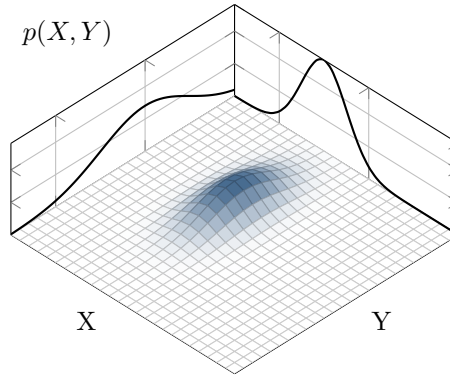
Why make it so complicated? This “trick” of thinking of two (or more) variables as one is actually very useful to deal with more complex cases (see below). It also makes clear that everything we know about probability distributions for single variables transfers directly to *joint* probability distributions for multiple variables. That includes positive definiteness and normalisation

$$p(X, Y) \stackrel{!}{\geq} 0 \quad \int_X \int_Y p(X, Y) \, dX \, dY \stackrel{!}{=} 1 \quad \sum_{X, Y} P(X, Y) \stackrel{!}{=} 1 \quad (7)$$

as well as the difference between probability *density* and probability *mass* functions. It now also comes more naturally to have mixed discrete-continuous domains or non-trivial regions to integrate over.

Example

A robot is throwing a ball. The location where the ball hits the ground is described by the distance X from the robot and a deviation Y perpendicular to the throwing direction. X and Y are jointly normally distributed as $p(X, Y)$:



Now, for any 2-dimensional region \mathcal{A} in the X - Y -plane, we can compute the probability of the ball landing within that region by integrating $p(X, Y)$ over \mathcal{A} .

Questions

- What do the equations in equation 7 state and how would you need to “fix” the short notation to be mathematically correct?

The three equations state positive definiteness, normalisation (continuous case), and normalisation (discrete case). The full notation would be:

$$\forall x \in \text{dom}(X) \forall y \in \text{dom}(Y) : p(X = x, Y = y) \stackrel{!}{\geq} 0 \quad (8)$$

$$\int_{\text{dom}(X)} \int_{\text{dom}(Y)} p(X = x, Y = y) \, dx \, dy \stackrel{!}{=} 1 \quad (9)$$

$$\sum_{x \in \text{dom}(X)} \sum_{y \in \text{dom}(Y)} P(X = x, Y = y) \stackrel{!}{=} 1 \quad (10)$$

- What is the probability of the ball landing no farther than a distance R from the robot? Just write down the integral, you do not need to solve it!

The region of interest \mathcal{A} is a circle of radius R around the robot at $(X, Y) = (0, 0)$

$$\mathcal{A} := \{(x, y) \in \text{dom}(X) \times \text{dom}(Y) : x^2 + y^2 \leq R^2\} . \quad (11)$$

We then have to integrate

$$P((X, Y) \in \mathcal{A}) = \iint_{\mathcal{A}} p(X, Y) \, dY \, dX , \quad (12)$$

where $P((X, Y) \in \mathcal{A})$ is the probability of the (binary) event of the ball landing within the circle.

2.3.2 Marginal and conditional probability distributions

What if two or more variables are jointly distributed, but we are only interested in the value of one (or a subset) of them? Maybe we already know the values of the other variables. Or we do not care what values they take. This is what *marginal* and *conditional* probability distributions are for. Mathematically, “not caring” about the value of a variable y corresponds to summing/integrating over all its possible values. Therefore, for a given *joint* distribution $p(x, y)$ of two variables x and y , the *marginal* distribution $p(x)$ is defined as

$$p(x) := \sum_y p(x, y) \quad \text{or} \quad p(x) := \int p(x, y) dy . \quad (13)$$

In contrast, if we already know the value of y and are interested in the distribution of x for any given value of y , this is described by the *conditional* distribution

$$p(x | y) := \frac{p(x, y)}{p(y)} . \quad (14)$$

Intuitively, $p(x, y)$ describes the probability of x and y both *by chance* taking a specific value. But since we already know the value of y , we need to divide by $p(y)$ (the probability of just y taking this specific value, independently of the value of x), since this “removes the amount of chance” that is too much in $p(x, y)$.

It follows one very important mathematical identity (called the *chain rule*) linking *joint*, *marginal*, and *conditional* probability distributions over the same variables:

$$\underbrace{p(x, y)}_{\text{joint}} = \underbrace{p(x | y)}_{\text{conditional}} \overbrace{p(y)}^{\text{marginal}} = p(y | x) p(x) . \quad (15)$$

Intuitively, you can think of this as follows: It does not matter whether we

- sample x and y both jointly at the same time from $p(x, y)$ or
- first sample y from $p(y)$ and then sample x from $p(x | y)$ or
- first sample x from $p(x)$ and then sample y from $p(y | x)$.

You can also turn it around by asking: How do we sample from the marginal $p(x)$ if we only have $p(x, y)$? Easy, you just sample from $p(x, y)$ and ignore the value of y . This essentially is the *definition* of a marginal probability distribution (you just add up all possibilities of sampling x).

Questions

- Rewrite equation 14 in log-representation, i.e., take the logarithm on both sides of the equation. Does the operation of “removing the amount of chance” make more sense in log-representation?

In log-representation

$$\log p(x | y) = \log \frac{p(x, y)}{p(y)} = \log p(x, y) - \log p(y) . \quad (16)$$

the log-probabilities are actually added and subtracted. Generally, log-representation is much more intuitive and natural in many respects.

- Prove that conditional probability distributions are properly normalised (use their definition equation 14 and the definition of the marginal equation 13).

Taking the sum over x on both sides of equation 14 and inserting equation 13 yields

$$\sum_x p(x | y) = \frac{1}{p(y)} \sum_x p(x, y) = \frac{1}{p(y)} p(y) = 1 . \quad (17)$$

- Recover equation 13 and equation 14 from equation 15.

For equation 13, we can take the sum over y on both sides of equation 15

$$\sum_y p(x, y) = \sum_y p(y | x) \underbrace{p(x)}_{\substack{\text{does not} \\ \text{depend on } y}} = p(x) \underbrace{\sum_y p(y | x)}_{=1 \text{ (normalisation)}} = p(x) . \quad (18)$$

For equation 14, we divide both sides of equation 15 by $p(y)$.

- Derive Bayes' formula

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} \quad (19)$$

from equation 15.

Divide both sides by $p(x)$.

- Using equation 15 and the “two variables as one” trick (c.f. equation 6), derive the chain rule for three variables, then for n variables.

For three variables x, y, z :

$$p(x, y, z) = \underbrace{p(x, a)}_{a:=(y,z)} = p(x|a)p(a) = p(x|y, z)p(y, z) = p(x|y, z)p(y|z)p(z) . \quad (20)$$

For n variables x_1, \dots, x_n :

$$p(x_1, \dots, x_n) = \underbrace{p(x_1, a)}_{a:=(x_2, \dots, x_n)} = p(x_1|x_2, \dots, x_n)p(x_2, \dots, x_n) \quad (21)$$

$$= p(x_1|x_2, \dots, x_n)p(x_2|x_3, \dots, x_n) \cdots p(x_{n-1}|x_n)p(x_n) . \quad (22)$$

- For the following joint distribution $p(x, y)$ compute the marginals $p(x)$ and $p(y)$ and check that all three normalisation requirements are satisfied.
 - Write the marginals in the... margins (right/bottom).
 - Use NumPy or PyTorch to do the same.

		y			
		$p(x, y)$	0	1	2
x	0	0.08	0.12	0.20	$p(x)$
	1	0.12	0.18	0.30	
			$p(y)$		

Answer:

		y					
		$p(x, y)$	0	1	2	3	
x	0	0.08	0.12	0.20	0.4	$p(x=0)$	
	1	0.12	0.18	0.30	0.6	$p(x=1)$	
	2	0.2	0.3	0.5	1.0		
			$p(y=0)$	$p(y=1)$	$p(y=2)$		

Listing 3: Computing marginals

```

1 pxy = [[0.08, 0.12, 0.20],
2        [0.12, 0.18, 0.30]]
3
4 pxy = np.array(pxy)      # NumPy version
5 pxy = torch.tensor(pxy)  # PyTorch version
6
7 py = pxy.sum(axis=0) # axis=0 is x-axis, so this is p(y)
8 py
9 -> [0.2, 0.3, 0.5]
10 py.sum()
11 -> 1.0
12
13 px = pxy.sum(axis=1) # axis=1 is y-axis, so this is p(x)
14 px
15 -> [0.4, 0.6]
16 px.sum()
17 -> 1.0
18
19 pxy.sum()
20 -> 1.0

```

- You have an unnormalised array $\begin{vmatrix} 0.8 & 1.2 & 2.0 \\ 1.2 & 1.8 & 3.0 \end{vmatrix}$. Use NumPy/PyTorch to
 - normalise it to get $p(x, y)$
 - normalise it to get $p(x|y)$
 - normalise it to get $p(y|x)$.

Double-check normalisation!

Listing 4: Computing conditionals

```

1 pxy_ = [[0.8, 1.2, 2.0],
2         [1.2, 1.8, 3.0]]
3
4 pxy_ = np.array(pxy)      # NumPy version
5 pxy_ = torch.tensor(pxy)  # PyTorch version
6
7 pxy = pxy_ / pxy_.sum()
8 pxy
9 -> [[0.08, 0.12, 0.20],
10     [0.12, 0.18, 0.30]]
11 pxy.sum()
12 -> 1.0
13
14 px_on_y = pxy_ / pxy_.sum(axis=0, keepdims=True) # normalised over x
15 px_on_y
16 -> [[0.4, 0.4, 0.4],
17     [0.6, 0.6, 0.6]]
18 px_on_y.sum(axis=0)                                # normalised for any value of y
19 -> [1., 1., 1.]
20
21 py_on_x = pxy_ / pxy_.sum(axis=1, keepdims=True) # normalised over y
22 py_on_x
23 -> [[0.2, 0.3, 0.5],
24     [0.2, 0.3, 0.5]]
25 py_on_x.sum(axis=1)                                # normalised for any value of x
26 -> [1., 1.]

```

- Follow-up questions on the code:
 - Do we need `keepdims=True`? Why yes/no?

We need it for $p(y|x)$ but not for $p(x|y)$. For $p(y|x)$ the two arrays have shapes $(2, 3)$ and $(3,)$, so they broadcast (since they match in the last dimensions). For $p(x|y)$ we have $(2, 3)$ and $(2,)$ instead, which does not broadcast. Using `keepdims=True` keeps the last dimension (which is summed over), so we get $(2, 1)$, which now *does* broadcast.
 - Why can we use the unnormalised version (`pxy_` in the solution) to compute the conditionals?

The normalisation factor of $p(x, y)$ is also a constant factor in $p(x)$ and $p(y)$, so it cancels out when computing the conditionals and we can leave it out from the beginning

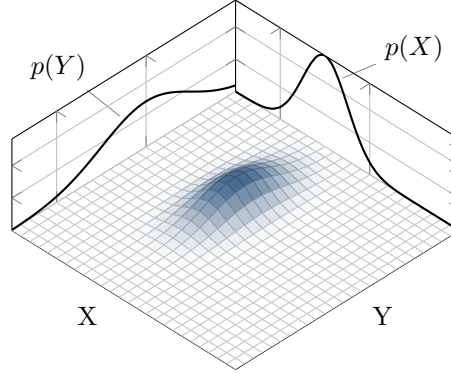
$$p(x | y) = \frac{p(x, y)}{p(y)} = \frac{\tilde{p}(x, y)/Z}{\tilde{p}(y)/Z} = \frac{\tilde{p}(x, y)}{\tilde{p}(y)} , \quad (23)$$

where \tilde{p} are the unnormalised versions and Z is the normalisation constant

$$Z = \sum_{x, y} \tilde{p}(x, y) . \quad (24)$$

Example (marginal)

The following plot shows a normal distribution $p(X, Y)$ long with its marginals $p(X)$ and $p(Y)$:



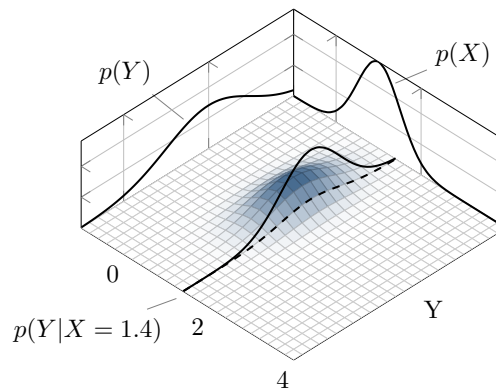
Generally, if you want to get rid of a variable from a joint distribution, you marginalise it out (unless you know its value, in which case you condition on it). This works with arbitrarily many variables. So, for example, if you have n variables x_1, \dots, x_n jointly distributed as $p(x_1, \dots, x_n)$, the marginal distribution of x_1 is found by integrating out all the other dimensions

$$\underbrace{p(x_1)}_{\text{for } x_1 \text{ alone}} = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(x_1, \dots, x_n) \underbrace{dx_2 dx_3 \cdots dx_n}_{\text{is found by integrating out } x_2 \text{ up to } x_n} . \quad (25)$$

So we have $n - 1$ integrals over the joint density function to get the marginal distribution of x_1 alone.

Example (conditional)

The following plot shows a normal distribution $p(X, Y)$ long with the marginals $p(X)$ and $p(Y)$, and the conditional $p(Y | X = 1.4)$:



The dashed line indicates $p(Y, X = 1.4)$, which is proportional to $p(Y | X = 1.4)$ but is not properly normalised. The solid line shows

$$p(Y | X = 1.4) = \frac{p(Y, X = 1.4)}{p(X = 1.4)} , \quad (26)$$

where $p(X = 1.4)$ acts as the normalisation constant. Generally, the conditional $p(Y | X)$ is a function of two variables (X and Y) just like the joint $p(X, Y)$. For each specific value of X they are proportional to each other, differing by the factor $p(X)$. Conditional distributions allow you to answer questions like “what is the probability of a persons age Y , given that their height X is 1.4m?”

3 Independence

If you are dealing with more than one variable, the different variables may *statistically depend* on each other – or not. This is very important to know, because it tells you what variables to take into account, e.g., in your agent’s policy. Why do you only need to consider the current state of the environment and not go further back in history?

Intuitively, two variables being statistically dependent on each other means that knowing the value of one tells you something about the other. Mathematically, this can be formulated by the relation between the marginals (where we ignore other variables) and the joint distribution (where we look at them simultaneously). Two variables x and y are *statistically independent*, written $x \perp\!\!\!\perp y$, if and only if their joint distribution equals the product of their marginal distributions

$$x \perp\!\!\!\perp y \iff p(x, y) = p(x) p(y) . \quad (27)$$

Generally, variables cannot be assumed to be independent, that is,

$$x \not\perp\!\!\!\perp y \iff p(x, y) \neq p(x) p(y) . \quad (28)$$

Sometimes, two variables *become* independent if we know the value of a third variable. This is called *conditional independence*:

$$x \perp\!\!\!\perp y \mid z \iff p(x, y \mid z) = p(x \mid z) p(y \mid z) . \quad (29)$$

Again, this is not generally the case and writing something like equation 29 is an additional *assumption* about your problem.

This makes clear that by marginalising a variable out, in general, we *lose information* and the joint distribution cannot be reconstructed from the individual marginals – unless the variables are independent. It is very important not to confuse two things:

1. factorisations that can *always* be made (mathematical identities), such as the chain rule in equation 15
2. factorisations that can only be made based on *additional independence assumptions*, such as in equation 27.

Questions

- Are the two variables described by this joint distribution (same as above) independent or not?

		y				
		$p(x, y)$	0	1	2	
x	0	0.08	0.12	0.20	$p(x)$	
	1	0.12	0.18	0.30		
						$p(y)$

Yes, they are! This can be verified by checking that for all possible assignments of x and y , the value of the joint $p(x, y)$ is actually the product of the values of the marginals $p(x)$ and $p(y)$:

		y					
		$p(x, y)$	0	1	2	3	
x	0	0.08	0.12	0.20	0.4	$p(x = 0)$	
	1	0.12	0.18	0.30	0.6	$p(x = 1)$	
	2	0.2	0.3	0.5	1.0		
			$p(y = 0)$	$p(y = 1)$	$p(y = 2)$		

- If you multiply both sides of equation 29 with $p(z)$ you get

$$p(x, y, z) = p(x | z) p(y | z) p(z) . \quad (30)$$

Compare that to the chain rule for three variables (see other question above). What is the difference?

The chain rule for three variables is

$$p(x, y, z) = p(x | y, z) p(y | z) p(z) , \quad (31)$$

where x additionally depends on y and not only z . Dropping this y is equivalent to assuming $x \perp\!\!\!\perp y | z$.

3.1 Graphical models

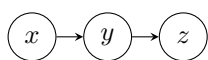


This is a very short excursion as graphical models are worth an entire lecture by themselves. The point is that in many cases it is important to understand the high-level structure of a problem and make this explicit. Generally, we need to know “what depends on what” and “what probability distributions are given”. This is precisely what graphical models do: they are a high-level visual language to describe dependencies (and more importantly independencies) between variables. We will only deal with *directed* graphical models (also called Bayesian networks). They are directed acyclic graphs with nodes that represent random variables and edges that denote conditional independence:

A variable is *independent* of all other variables *conditional* on its incoming neighbours (i.e., all variables “pointing” to it).

Thus, a graphical model encodes assumptions about our problem and tells us what dependencies we can drop. Using the assumptions encoded in a graphical model, the joint distribution over n variables can be written (factorised) as a “simplified chain rule” with n conditional distributions, where each variable is conditioned only on its incoming neighbours.

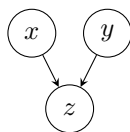
Examples



$$p(x, y, z) \equiv \overbrace{p(z | y, x) p(y | x) p(x)}^{\text{always possible (chain rule)}} \quad (32)$$

$$= \underbrace{p(z | y) p(y | x) p(x)}_{\text{dropping } x \text{ makes an independence assumption}} \quad (33)$$

dropping x makes an independence assumption



$$p(x, y, z) \equiv p(z | y, x) p(y | x) p(x) \quad (34)$$

$$= p(z | x, y) p(y) p(x) \quad (35)$$

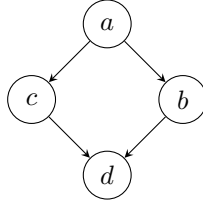


$$p(x_1, x_2, \dots, x_{n-1}, x_n) \equiv p(x_n | x_{n-1}, x_{n-2}, \dots, x_2, x_1) p(x_{n-1} | x_{n-2}, \dots, x_2, x_1) \dots p(x_2 | x_1) p(x_1) \quad (36)$$

$$= p(x_n | x_{n-1}) p(x_{n-1} | x_{n-2}) \dots p(x_2 | x_1) p(x_1) \quad (37)$$

Question

- How can the joint distribution $p(a, b, c, d)$ be factorised and simplified given the independence assumptions encoded in the following graphical model:



$$p(a, b, c, d) = \overbrace{p(d | \mathbf{a}, b, c) p(c | a, \mathbf{b}) p(b | a) p(a)}^{\text{normal chain rule}} \quad (38)$$

$$= \underbrace{p(d | b, c) p(c | a) p(b | a) p(a)}_{\text{simplified}} \quad (39)$$

4 Expectations

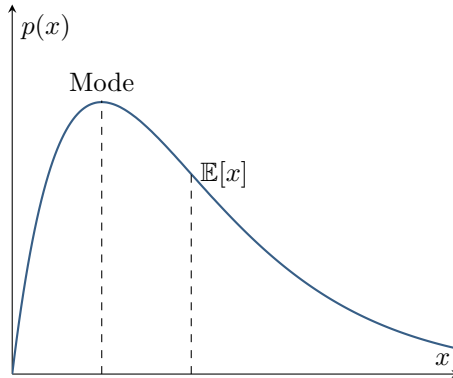
The expected value or mean value for a random variable x is calculated by integrating/summing over the product of the variable's value and the corresponding probability density/mass $p(x)$

$$\mathbb{E}_{p(x)}[x] = \int x p(x) dx \quad \mathbb{E}_{p(x)}[x] = \sum_x x p(x) . \quad (40)$$

Similarly, the expected value for any measurable function of a random variable is computed by integrating/-summing the product of the function value and the density/mass

$$\mathbb{E}_{p(x)}[g(x)] = \int g(x) p(x) dx \quad \mathbb{E}_{p(x)}[g(x)] = \sum_x g(x) p(x) . \quad (41)$$

It is important to see that the mean is not the same as the mode (the value with highest probability). This is particularly important for asymmetric distributions (see plot below) and multi-modal distribution (distribution with more than one local maximum).



If you think of the distribution of all possible faces, you can think of the mode as the most probable sample, a hyper realistic face of the most likely person. The expectation, on the other hand, is the average face, a blurry and improbable face.

Why do we care about expectations in reinforcement learning?

- Well, going back to our example of a population of people in Durham whose heights follow a normal distribution with mean $\mu = 178\text{cm}$ and standard deviation $\sigma = 8\text{cm}$, let's imagine we have an agent that tries to guess the height of each person before seeing them, where everyone is standing in a random queue and it can only observe one person at a time.
- While sometimes the agent's height prediction will be wrong, e.g., if the queue had lots of small children, with enough averaging it can update its predictions to converge to the correct answer.
- More formally, this is $\mathbb{E}_{x \sim N(\mu, \sigma^2)}[x] = \mu$, which is that the expected value of heights x sampled from a normal distribution is equal to the mean.

- In other words, we use expectations to average the results as our agent interacts with the environment to assess the value of the states, then we improve our policy based on those averaged values.

Common notation:

- Often in reinforcement learning papers you will see notation like $\mathbb{E}_{\mathbf{x} \sim P}[f(\mathbf{x})]$, which reads as the “expectation of $f(\mathbf{x})$ with respect to $P(\mathbf{x})$ ”. This means drawing sample(s) from some unknown data distribution (sampling the dataset). Sometimes p_{data} is used instead of $P(x)$.
- For example $\mathcal{L} = \mathbb{E}_{\mathbf{x} \sim P}[||f(\mathbf{x}) - \mathbf{y}||^2]$ would be the average error of the squared L2 norm between the function $f(\mathbf{x})$ and the targets \mathbf{y} . This would be averaging the error after evaluating the function (agent) across many samples from the dataset.

Simple example:

- What is the expected value of throwing an unbiased six-sided die?
 - A die has a probability mass function where there is equal probability $\frac{1}{6}$ of taking on each of the six values—a discrete uniform distribution.
 - So the expected value is computed by evaluating the value x_i times the probability $p(x_i)$ which is just $1/6$

$$\begin{aligned}\mathbb{E}[x] &= \sum_{i=1} x_i p(x_i) \\ &= 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} \\ &= \frac{1 + 2 + 3 + 4 + 5 + 6}{6} \\ &= 3.5\end{aligned}$$

Question

- What is the expected value of a biased six-sided die with probabilities (0.1, 0.1, 0.1, 0.1, 0.1, 0.5)?

$$\mathbb{E}[x] = \sum_{i=1} x_i p(x_i) \tag{42}$$

$$= 0.1(1 + 2 + 3 + 4 + 5) + 0.5 \cdot 6 = 4.5 . \tag{43}$$

- Write a program to evaluate the expected value of a die with an unknown distribution (unknown weights) from a sample of die throws.
 - Advanced: can you implement this without knowing how many die throws you need to make? Hint, it can be achieved using an incremental mean.

$$\mu_k = \frac{1}{k} \sum_{j=1}^k x_j \tag{44}$$

$$= \frac{1}{k} \left(x_k + \sum_{j=1}^{k-1} x_j \right) \tag{45}$$

$$= \frac{1}{k} (x_k + (k-1)\mu_{k-1}) \tag{46}$$

$$= \mu_{k-1} + \frac{1}{k} (x_k - \mu_{k-1}) \tag{47}$$

$$\approx \mu_{k-1} + \alpha (x_k - \mu_{k-1}) \tag{48}$$

$$= (1 - \alpha)\mu_{k-1} + \alpha x_k \tag{49}$$

For a smoothing average, you can replace $\frac{1}{k}$ with some small learning rate α , e.g. $\alpha = 0.001$.

Listing 5: Answer

```
1 mu = 0
2 unknown_probs = [0.1, 0.1, 0.1, 0.1, 0.1, 0.5]
3
4 for i in range(5000):
5     x = np.random.choice(len(unknown_probs), 1, p=unknown_probs)[0]
6     mu = mu + 0.001 * (x - mu)
7
8 print(mu+1) # dice numbers start from 1
```
