CO6: Introduction to Computational Neuroscience

Lecturer: J Lussange
Ecole Normale Supérieure
29 rue d'Ulm
e-mail: johann.lussange@ens.fr

Solutions to the 2nd exercise sheet

If you have any questions regarding these solutions or the exercises in general, feel free to contact me by e-mail (florian.dehmelt@ens.fr).

Math reminder

Although you all know what a vector is, many of you didn't come across one in the last couple of years. So here's a little reminder on their most essential properties:

First of all, a vector is nothing but a fixed arrangement of values, that is: a way to write down a large number of things using a single symbol. For example, if you want to come up with a formula containing all of the internal estimates m_i a bee may have about the value of N different flowers, then instead of writing these estimates down indivually, you may instead want to use the following vector:

$$\mathbf{m} = \begin{pmatrix} m_1 \\ m_2 \\ \vdots \\ m_N \end{pmatrix}$$

Although one may use a vector in order to denote a position in space (that is: the x, y and z coordinates of a certain point) and sometimes even use arrows to represent them, the concept of vectors is by no means limited to that - and a vector may in fact contain anything you want it to: positions, times, expected reward values... or all of these at once.

Whereas in international literature, vectors are usually denoted by **bold** letters, these are rather difficult to reproduce in handwriting. Instead, you may want to use one of the many alternative conventions:

$$\mathbf{m}\,, \quad \vec{m}\,, \quad \underline{m}\,, \quad \ldots$$

It is quite helpful to make a clear distinction between vectors and scalars (that is: simple numbers) in order to avoid confusion.

For now, you only need to remember three basic vector operations: how to add or subtract two vectors, how to multiply them with each other, and how to multiply one of them with a scalar (a number).

If we consider two N-dimensional vectors **a** and **b**, we can add them in the following way:

$$\mathbf{a} + \mathbf{b} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix} = \begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \\ \vdots \\ a_N + b_N \end{pmatrix}$$

Multiplying a vector **a** with a scalar α yields

$$\alpha \cdot \mathbf{a} = \alpha \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} \alpha \cdot a_1 \\ \alpha \cdot a_2 \\ \vdots \\ \alpha \cdot a_N \end{pmatrix}$$

and the multiplication of two vectors **a** and **b** is defined as:

$$\mathbf{a} \cdot \mathbf{b} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix} = \sum_{k=1}^{n} a_k b_k$$

$$= a_1 b_1 + a_2 b_2 + \ldots + a_N b_N$$

Please note that, mathematically speaking, 'vertical' and 'horizontal' vectors (also referred to as 'column vectors' and 'line vectors') are two different things. Strictly speaking, if you have a column vector and you want to 'flip it around' in order to write it onto a single line, you would have to indicate this flipping around (or: 'transposition') by adding the letter T:

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix} = \left(a_1,\ a_2,\ \cdots,\ a_N\right)^{\mathsf{T}} = \left(\mathbf{a}^{\mathsf{T}}\right)^{\mathsf{T}}$$

$$\neq \left(a_1,\ a_2,\ \cdots,\ a_N\right) = \mathbf{a}^{\mathsf{T}}$$

On many occasions, though, if it is clear from the context which of the two we are referring to, the difference between vectors and their corresponding transposed vectors is neglected. In fact, the product of two vectors shown above could be written as $\mathbf{a}^{\mathsf{T}}\mathbf{b}$ instead of $\mathbf{a} \cdot \mathbf{b}$.

A second issue I would like to remind you of is the correct use of logical symbols instead of using words such as 'thus', 'which leads to' and the like. This is particularly important when you derive a result step by step from certain initial assumptions.

For example, the equivalence symbol '$\Leftrightarrow$' should only be used if two expressions really are equivalent. They are equivalent, if the one on its right side is always true when the one on its left side is true, *and vice versa*. For example,

$$a^2 = 4 \quad \Leftrightarrow \quad a = 2$$

is *not* correct, whereas

$$a^2 = 4 \quad \Leftarrow \quad a = 2$$

is perfectly fine. Some of you may also have been confused by symbols like '$\rightarrow$' or '$\leftarrow$' frequently used in learning rules. Remember that these are *not* equivalent to the equality sign '$=$'. Instead, they symbolically illustrate the change of a value from one time step to the next. Thus, the following two expressions are equivalent:

$$a \ \rightarrow \ a + b \quad \Leftrightarrow \quad a_{\text{ new}} = a_{\text{ old}} + b$$

Again, if you have any questions regarding these or other mathematical concepts, feel free to contact me by e-mail. I will then try to cover these topics during the next seminar.

Attribution of points

For each exercise sheet, you can obtain up to 20 points, plus a variable number of bonus points for the exercises labeled as 'advanced'.

This time, points were attributed as follows:

exercise	points
1	10
2a	10
2b	+ 4

In these exercises, I have paid particular attention to your way of explaining what you're doing, and why you're doing it. Which is why 'similar' solutions may have received different grades.

Discounted TD learning Exercise 1

Let us first have another look at non-discounted temporal difference learning, and how we deduced the corresponding learning rule during Christian's lecture.

Imagine an animal which experiences a sequence of states:

$$\ldots \rightarrow s \rightarrow s+1 \rightarrow s+2 \rightarrow \ldots$$

These states may be intersections of a maze, in which the animal has to decide between possible actions, or just a sequence of stimuli which are being presented without any action taken by the animal itself.

In either case, we can define how valuable a particular state is with respect to some measure we choose. For example, reaching a particular state s might lead to a reward $r(s)$ being delivered in this state. Furthermore, this initial state might lead to other states $s+1$, $s+2$ etc. which are also being rewarded. We may thus decide to consider a state to be all the more valuable, the higher the total reward is which the animal receives for being in this state and in all the states that follow it.

Thus, we can define the value of a state as

$$\begin{aligned} V(s) &= r(s) + r(s+1) + r(s+2) + \ldots \\ &= \sum_{\tau=0}^{\infty} r(s+\tau) \end{aligned} \tag{1}$$

It is worth noting that this definition of our value function does not make any difference between immediate rewards and rewards in the distant future. This is why we will eventually move on to a fancier model including temporal 'discounting' at the end of this exercise.

But for now, let us continue with this simple model. We have just defined how valuable a particular state *really is* - but the animal does not know this value! All the animal has is an internal *estimate* of how valuable all these possible states might be. Initially, these estimates may be totally wrong, but as the animal keeps on learning, they will gradually improve until, eventually, they approach the *real* value of each and every state.

A simple approach to learning is to assume that whenever the animal has completed a trial (for example, has run all its way through a maze and subsequently been be taken out), it will reconsider the different states it encountered during the trial, as well as the rewards associated with them. By evaluating how well its previous estimates of the value of each state correlated with the value it experienced, it will then update each estimate to a new value:

$$\hat{V} \rightarrow \hat{V} + \epsilon\delta \tag{2}$$

Here, δ is the prediction error, and ϵ is a constant which determines how quickly the animal updates its estimates. There are different ways to define δ, of which the following one is the most obvious:

$$\delta = V(s) - \hat{V}(s)$$

In other words, the prediction error which the animal made when it last reached state s is the difference between the real value of that state and the value the animal *believed* it had.

This makes perfect sense, but there is a problem - the animal is unable to compute such an error, precisely because it does not know the *real* value of the state!

Hence, the animal must be doing something else. Which is why people came up with the idea that it might just be making the following approximation:

$$\begin{aligned} \delta &= V(s) - \hat{V}(s) \\ &= r(s) + V(s+1) - \hat{V}(s) \\ &\approx r(s) + \hat{V}(s+1) - \hat{V}(s) \end{aligned} \tag{3}$$

In other words, the only external information the animal takes into account when updating

its internal estimates is the reward it received in the different states s it encountered. It then takes this information and compares it to the estimates it had previously made of the values of these states s and the states $s+1$ which would immediately follow it. This may not yield the precise error the animal *really* made - but it might still be quite close to it. And, importantly: the animal is actually able to do this!

But when exactly does the animal perform these updates of its internal estimates? Our initial approach was to assume that this evaluation takes place after the completion of every trial; and in his lecture, Christian demonstrated how this can yield some rather satisfying results (think of Rescorla-Wagner and the like). But does this assumption really make sense?

As well as it may work in artificial, 'Pavlovian' paradigms, it is hard to see how you could implement such a kind of learning in 'real life'. In most if not all of the situations an animal encounters in the wild, there won't be a repetition of discrete episodes. Or at least, what you might consider an 'episode' can be quite long, and it would certainly be desirable to start improving your performance *during* this episode.

So instead of waiting for the end of each episode before applying its learning rule (**??**), the animal might do so after every time step! Which is what defines 'temporal-difference learning' and distinguishes it from pure Monte-Carlo approaches.

Please note that this does not necessarily imply that TD learning is faster than other kinds of learning; it just happens in a different way.

What about the actual exercise?

So much for the general idea of TD learning. Now what about the exercise you had to solve? If we want to obtain an explicit version of the learning rule in the non-discounted case, we simply have to combine our definition of value (**??**) with our learning rule (**??**) and perform the approximation shown in equation (**??**):

$$\hat{V} \to \hat{V} + \epsilon \cdot \Big(r(s) + \hat{V}(s+1) - \hat{V}(s) \Big)$$

This is the learning rule if we equally cherish all rewards, no matter *when* they are obtained. If we want to take into account that immediate rewards can be much more gratifying than rewards in the distant future, we can introduce a discounting parameter γ (which may be anything between 0 and 1). For every time step separating a potential reward from the state of the animal which we want to evaluate, the contribution of this reward is discounted by multiplying it with γ. Our value function (**??**) then becomes:

$$\begin{aligned} V(s) &= r(s) + \gamma r(s+1) + \gamma^2 r(s+2) + \dots \\ &= \sum_{\tau=0}^{\infty} \gamma^\tau r(s+\tau) \end{aligned} \tag{4}$$

And this is the only change we need to make! Our generic learning rule (**??**) is still applicable, and we can simply substitute our new value function into it. Because

$$\begin{aligned} V(s) &= r(s) + \gamma r(s+1) + \gamma^2 r(s+2) + \dots \\ &= r(s) + \gamma \left[r(s+1) + \gamma r(s+2) + \dots \right] \\ &= r(s) + \gamma \sum_{\tau=0}^{\infty} \gamma^\tau r(s+1+\tau) \\ &= r(s) + \gamma V(s+1) \end{aligned}$$

we can make a similar approximation as in equation (**??**):

$$\begin{aligned} \delta &= V(s) - \hat{V}(s) \\ &= r(s) + \gamma V(s+1) - \hat{V}(s) \\ &\approx r(s) + \gamma \hat{V}(s+1) - \hat{V}(s) \end{aligned}$$

All we need to do now is to substitute this into our generic learning rule (**??**) to obtain:

$$\hat{V} \to \hat{V} + \epsilon \cdot \Big(r(s) + \gamma \hat{V}(s+1) - \hat{V}(s) \Big)$$

Which is the learning rule for discounted temporal difference learning. Ça y est.

Model-based evaluation

What we have seen so far was how an animal can learn the value of certain stimuli - or combinations of stimuli - after being exposed to them a number of times. In 'real life' though, most of these stimuli will appear in an infinite number of combinations with other stimuli we already know. In a model-free approach, we would then have to learn each one of these combinations from scratch.

Remember that 'value' can be anything of biological relevance - from quantities of nectar obtained by a bee to the intensity of fear induced in a rat due to the presence of one or more cats (aversive stimulus). In the latter case, slowly learning how to deal with a new situation could be particularly harmful.

This is where model-based evaluation policies come in. Here, a 'model' is a rule which tells the animal how to *generalize*, that is: to evaluate a combination of different stimuli, based on the value the animal had previously attributed to each individual stimulus it comprises.

One can imagine many different ways of implementing this. A very simple way to do so is to assume this model to be *linear*. 'Linearity' implies that all the different elements which contribute to the total value of a state do so independently of each other. Which is very convenient, because it allows us to determine their contributions individually - if we then want to compute the total value, we can simply add them up.

So if the animal attributes values $\hat{V} = w_1$ and $\hat{V} = w_2$ to stimuli 1 and 2 whenever they are presented individually, it will expect the following value if both stimuli suddenly appear together:

$$\hat{V} = w_1 + w_2 \tag{5}$$

If you think of our poor little rodent, this would mean that two cats were twice as scary as one of them alone. Which is not necessarily the case, because the two cats might team up to apply some advanced hunting tactics... which is really, really scary for our rat. And which, maybe, should rather be modelled by *multiplying* the contributions of each cat to overall scaryness instead of just adding them.

But for now, let us look at a purely linear model of our value function.

Now, if we want to express our total value function in a way we can handle no matter which of the two stimuli is present or absent, we can write:

$$\hat{V} = w_1 \cdot u_1 + w_2 \cdot u_2 \tag{6}$$

Here, u_i denotes the presence ($u_i = 1$) or absence ($u_i = 0$) of the respective stimulus. If both stimuli are present ($u_1 = u_2 = 1$), we still obtain equation (**??**).

If the number of stimuli which could potentially be combined is very large, writing our model as a summation, as we did in equation (**??**), can be quite lengthy. Thus, for combinations of N different stimuli, we will instead introduce vectors to describe the contribution to total value of the individual stimuli (that is, their *weights* w_i) and their presence or absence u_i:

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}$$

Thus, vector **w** describes the linear model the animal has of all the stimuli in the world around it, whereas vector **u** describes one particular situation the animal may encounter. Combining the two then yields the expected value of that situation:

$$\hat{V}(\mathbf{u}) = \mathbf{w} \cdot \mathbf{u} = \sum_{k=1}^{N} w_i u_i \tag{7}$$

$$= w_1 u_1 + w_2 u_2 + \ldots + w_N u_N$$

Please note that in his lecture, Christian sometimes uses different notations. He prefers not to speak of the 'real value' V of a state, but to refer to it as the 'experienced total future reward' R. This is why he is free to - and often does - denote estimates like $\hat{V}$ by a simple V.

Exercise 2a

In this exercise, the vector containing the weights of our linear model is not explicitly mentioned. But we do know which value the animal attributes to two different situations $\mathbf{u} = (1, 0)$ and $\mathbf{u} = (0, 1)$. From this, we can conclude what the weight vector actually looks like:

$$\begin{aligned}\alpha = \hat{V}\big(\mathbf{u} = (1,0)\big) &= \hat{V}(1,0) \\ &= w_1 \cdot 1 + w_2 \cdot 0 = w_1\end{aligned}$$

$$\begin{aligned}\beta = \hat{V}\big(\mathbf{u} = (0,1)\big) &= \hat{V}(0,1) \\ &= w_1 \cdot 0 + w_2 \cdot 1 = w_2\end{aligned}$$

Hence, the weight vector is:

$$\mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

We can now use this to calculate the value attributed to a situation in which both stimuli are present:

$$\hat{V}(1,1) = \alpha \cdot 1 + \beta \cdot 1 = \alpha + \beta$$

And for a fifty-percent chance of stimulus 1 being present, we obtain:

$$\hat{V}(0.5, 0) = \alpha \cdot 0.5 + \beta \cdot 0 = \frac{\alpha}{2}$$

The fact that this value is exactly half the value of stimulus 1 always being present (or in other words: that it is halfway between the values for $u_1 = 1$ and $u_1 = 0$) is of course due to the fact that we are looking at a purely *linear* model. For a real rat, the potential presence of a cat might be almost as scary as its actual presence.

Advanced exercise: Exercise 2b

Whereas in exercise 2a, we had assumed the weight vector to be constant, it will of course change over time if we allow for learning to take place. This is particularly important if we want to deal efficiently with a new stimulus being combined with a set of well-known stimuli.

In perfect analogy with equation (**??**), we can come up with the following learning rule for the individual weights:

$$w_i \to w_i + \epsilon\delta \cdot u_i$$

The additional factor u_i makes sure that a weight is only changed if its corresponding stimulus has been presented.

But why do we discuss the learning of **w**, when in fact we want to optimize $\hat{V}$? Remember that the weight vector **w** comprises all parameters of our explicit model (**??**). We can thus improve the performance of the model for V by simply improving the components w_i of the vector.

If we want to evaluate how good our estimate has become, we need to compare the value V computed from the current weight w_i to the 'real' value of that state. Thus, our prediction error δ will look the same as in exercise 1: It is the difference between the total future reward from state s onwards and the value the animal had previously attributed to that state:

$$\delta(s) = V(s) - \hat{V}(s)$$

We can then make the same approximation as in equation (**??**) to obtain:

$$\begin{aligned}\delta(s) &= r(s) + V(s+1) - \hat{V}(s) \\ &\approx r(s) + \hat{V}(s+1) - \hat{V}(s)\end{aligned}$$

To end up with the following learning rule:

$$w_i \to w_i + \epsilon u_i(s) \cdot \Big(r(s) + \hat{V}(s+1) - \hat{V}(s) \Big)$$

Given the model (**??**) our animal uses to estimate $\hat{V}$, this is equivalent to:

$$w_i \to w_i + \epsilon u_i(s) \cdot \Big(r(s) + \mathbf{w} \cdot \big[\mathbf{u}(s+1) - \mathbf{u}(s) \big] \Big)$$

If we associate each state with a point in time, we may also rewrite this equation accordingly, replacing 's' by 't' and '$s+1$' by '$t+\tau$'.