We’ve discussed normative frameworks for sensory coding, inference, and action selection. We’re now going to connect elements of each of these frameworks within the context of a problem that nearly all animals need to solve: foraging. This problem is inspired by work from Montague, Dayan, Person, and Sejnowski [1]. It also brings in elements from work by Laughlin [2] and DeWeese and Zador [3]. You can find these papers (and more) at the end of the problem set.

1 Problem setup

Consider a bee foraging in an environment that consists of patches of forests and fields. Each type of patch contains different types of flowers with different amounts of nectar. The colors of the flowers (and the colors alone) predict how much nectar they produce. We’ll parameterize this color using the frequency of light (i.e., purple flowers are signaled by higher frequencies / lower wavelengths of light). The forests contain mostly blue and purple flowers, while the fields contain mostly red and yellow flowers (Fig 1A). If we were to count the number of flowers of a given color frequency in each type of patch (forest versus field), we would find two distinct but overlapping histograms (Fig 1B). In the forests, the more purple the color (i.e. the higher the frequency), the more nectar a flower will have on average. In the field, it’s the opposite; lower frequencies yield more nectar (Fig 1C).

The bee’s goal is to collect as much nectar as possible. In order to do this, the bee has to learn (through its own experience) which flowers are more likely to produce nectar, and it has to learn how to change its
behavior in order to exploit this. Because there is a different relationship between flower color and reward in the forest versus the fields, the bee has to learn all of this in a context-dependent manner.

We’re going to take a very particular type of approach to this problem. Rather than assuming that the bee has unlimited computing power and infinite training data, as might be a more typical machine learning approach, we’re going to explicitly consider the constraints that biological systems have to overcome in solving this type of problem. Specifically, we’ll assume the following:

(i) the bee does not have perfect color acuity; there is only a limited range of colors that it can distinguish at any one time.

(ii) the bee does not explicitly know whether it’s in the field or the forest; it has to infer this from the pattern of flowers that it sees.

(iii) the bee doesn’t know which flowers produce nectar; it has to learn this through it’s own experience in an on-line fashion.

We’ll work through this problem in steps. We’ll first consider how the bee’s limited acuity could impact its color perception (Section 2). We’ll then consider how the bee could use this limited color information to infer whether it is in the forest or the field (Section 3). Finally, we’ll consider how the bee could learn to associate this color information with different outcomes (the presence or absence of nectar), and change its behavior accordingly (Section 4). Each section will be broken down into a series of shorter problems that you can work out using pen and paper. At the end of each section, you’ll find extra problems (marked with *) that are more involved, and often require some coding; please work though the entire set of non-starred problems before returning to these starred problems.

2 Sensory Coding

For simplicity, we’ll assume that the bee can use only a single “frequency detector” to encode incoming light signals. This detector transforms these signals into discrete spike counts via a saturating nonlinearity, as shown in Fig 2A. It’s easiest to think of this transformation in two steps: first, an incoming frequency is transformed via a continuous nonlinearity, and then the output of this nonlinearity is discretized into a small number of spike counts. You can then imagine “tuning” this nonlinearity by adjusting its slope and offset (upper and lower panels in Fig 2B, respectively), prior to the discretization step. Note that the discretization makes this a lossy transformation, which means that some information about the incoming frequency value is lost when converting to a spike count.

![Figure 2: A simplified frequency detector.](image)

2.1 First, consider the role of the continuous nonlinearity shown in Fig 2A. If the bee is to use this single (fixed) nonlinearity for both the forests and the fields, how should it be positioned along the frequency axis
to give the bee the best discriminability of incoming frequency signals? Use Fig 3A to sketch what you’d intu- 
vitively expect this to look like, based on the distributions of frequencies in the forests and fields [in Problem 2.3 we’ll revisit this from the perspective of the spike count distribution]. How would this change if the detector only had to function in the forests? Only in the fields? Sketch this in Fig 3B.

2.2 Now, consider the impact of the discretization into spike counts. Consider a detector that produces a minimum of 0 spikes and a maximum of 5 spikes. Use the forest-only nonlinearity that you drew in Fig 3B to sketch the distribution of spike counts that this detector would produce in the forest. Sketch this in Fig 3C.

2.3 What would you expect this distribution of spike counts to look like if its entropy were maximized? Sketch this in Fig 4A. [Note that maximizing the entropy of this distribution is equivalent to maximizing the information that the detector’s output (spike counts) conveys about its input (frequency values) in the absence of noise; for more background, check out Jorge Menéndez’s primer on info theory [4]]. Work backwards from this maximum-entropy distribution to draw the forest-only nonlinearity that would produce this distribution; sketch this in Fig 4B. Assuming that the input distributions in Fig 1B are Gaussian, what mathematical function should the nonlinearity take in order to guarantee this output distribution? In light of this answer, revisit your sketch in Fig 3A, and sketch the optimal nonlinearity for maximizing entropy given the distribution of frequencies in the forest and the fields.

2.4 Use Fig 4C to sketch the distribution of outputs that this detector (which was optimized for the forest) would produce in the field. What is the entropy of this distribution? Now think about this result in the context of adapting to changes in the environment. If this detector was “adapted” to the forest (i.e., optimized as in the previous problem) when the bee suddenly moves from the forest to the field, what is the challenge with adapting to this change?
Parametrize the nonlinearity using a sigmoidal function and discretize its output between 0 and a maximum spike count of \( k_{\text{max}} \). Numerically, find the optimal parameters of the nonlinearity that maximize the information that the output spike counts convey about the input frequency in either the forest or the field (you can assume that the distributions in Fig 1B are Gaussian with different means but the same variance). How do these parameters change when optimized for the forest versus the field? How would your results change if the distributions had different variances but the same mean?

3 Inferring Context

For the purposes of this next section, we’ll refer to the field and forest as two different “contexts”. We will consider how the bee should infer which context it is in based on the colors of the flowers that it sees. We’ll refer to the forest as “context A”, and the field as “context B” (Fig 5). At any given time, the bee can only be in one of these two contexts. We are going to derive an expression for the probability that the bee is in context A, given its sequence of flower encounters. We’ll call this quantity \( p_{\text{belief},t}^A \). (This is shorthand notation for \( p_{\text{belief},t}^A = p(C_t = A | f_{\tau \leq t}) \), which can be read as “the probability that current context is A, given the history of past flower encounters up to and including time \( t \)”). This probability will be updated over time based on the dynamics of context switches, and on the history of flower encounters. We’ll refer to this entire process of constructing and updating this probability as the “inference process”.

![Figure 5: Simplified setup for inferring different contexts.](image)

3.1 Write an expression for \( p_{t}^B \) in terms of \( p_{t}^A \).

3.2 As you can see from Fig 5A, roughly half of the environment consists of context A, and roughly half is in context B. Let’s assume that the bee “knows” this already, and that this knowledge is summarized in its “belief” \( p_{\text{belief},t=0}^A \) at time \( t = 0 \). Write down the values of \( p_{\text{belief},t=0}^A \) and \( p_{\text{belief},t=0}^B \).

3.3 Let’s imagine that the bee flies straight through the environment at a roughly constant speed. We will think of timesteps in terms of flower encounters (i.e., one timestep = one encounter), and we’ll assume that on average, the bee will encounter \( \sim 10 \) flowers in one patch before crossing the boundary into another patch (i.e., the bee encounters \( \sim 10 \) flowers per context, on average). What is the average probability of switching contexts? We’ll call this \( p_{\text{switch}} \), and for the remainder of the problem, we’ll assume that the bee knows this value.
We can represent these context switches using a graphical model that describes the possible contexts (circles in Fig 5B), and the probability of staying within versus switching between contexts (arrows in Fig 5B). Use your value of $p_{\text{switch}}$ to fill in the missing probabilities in Fig 5B.

### 3.4
You should have written above that the bee begins with a belief of $p_{\text{belief}, t=0} = 1/2$ that it is in context A. Now let’s consider what happens at time $t = 1$, when the bee encounters its first flower. We’ll break this down into two parts: (1) the probability that the bee is still in context A, given that it encountered a flower, regardless of its color, and (2) the probability that the bee is still in context A, given that it observed the specific color of the flower (in Bayesian filtering, these are referred to as the “prediction” and “update” steps, respectively; for more background, check out Anna Kutschireiter’s guide to nonlinear filtering [5]). We’ll tackle the prediction step here, and the update step in the following problem.

The probability $p_{\text{predict}, t=1}^A$ that the bee is in context A after encountering a flower (but not yet observing its color) can be written as:

\[
\begin{pmatrix}
\text{probability that the bee is in context A at time } t=1, \text{ after encountering a flower but before observing its color} \\
\text{probability that the bee was in context A at time } t=0
\end{pmatrix} = \begin{pmatrix}
\text{probability that the bee didn’t cross into a different patch} \\
\text{probability that the bee was in context B at time } t=0
\end{pmatrix} \times \begin{pmatrix}
\text{probability that the bee did cross into a different patch}
\end{pmatrix} + \begin{pmatrix}
\text{probability that the bee was in context B at time } t=0
\end{pmatrix} \times \begin{pmatrix}
\text{probability that the bee did cross into a different patch}
\end{pmatrix}
\]

Fill in all of these quantities using expressions that we have defined above, and then evaluate this expression to get a single number (the probability that the bee is in context A at time $t=1$, after encountering a flower but before observing its color). Did this probability increase, decrease, or stay the same from $t=0$ to $t=1$? In this case, when $p_{\text{belief}, t=0}^A = 1/2$, the bee initially believes the context is equally likely to be either A or B (i.e., the bee has maximum uncertainty about the current context). How would your results change if $p_{\text{belief}, t=0}^A = 1/4$? What happens if $p_{\text{switch}}$ doubles (i.e., the environment is more volatile, and the context changes more quickly)?

### 3.5
After observing the color of the flower, the bee can use this additional information to update its belief about the current context at time $t=1$, (i.e. to construct $p_{\text{belief}, t=1}^A$). We know the distribution of flower colors when the bee is in context A, and similarly when the bee is in context B (these are the distributions in Fig 1B). As before, let’s assume that each of these distributions is a Gaussian with the same variance, but different means. I.e., in each context, the distribution of flower colors can be written as:

\[
p(f|A) = \mathcal{N}(f; \mu_A, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(f - \mu_A)^2}{2\sigma^2}\right);
\]

\[
p(f|B) = \mathcal{N}(f; \mu_B, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(f - \mu_B)^2}{2\sigma^2}\right).
\]

Now let’s assume that the bee encounters a flower of frequency $f_1$ at time $t = 1$. We would say that the bee has gathered some additional evidence about its current context, and it can use this evidence to update its belief about the current context:

\[
\begin{pmatrix}
\text{probability that the bee is in context A at time } t=1, \text{ after encountering a flower} \\
\text{probability that the bee is in context A at time } t=1, \text{ after encountering a flower but before observing its color}
\end{pmatrix} \propto \begin{pmatrix}
\text{probability that the bee would encounter a flower } f_1 \text{ if it is in context A}
\end{pmatrix}
\]

Fill in all of these quantities using expressions that we have defined above. *Bonus*: How would you remove the proportionality in the expression above, in favor of an equality?
3.6 The belief $p_{\text{belief}, t}^A$ summarizes the bee’s knowledge about the current context at time $t$. Over time, the bee can continue to update this belief as it encounters more flowers. After each flower encounter, this belief is used to form a prediction about the next timestep (as worked out in Problem 3.4), which is then updated when the bee observes the flower color (as worked out in Problem 3.5). In this way, the bee uses knowledge of the dynamics of the context switches and the values of each flower encounter to incrementally update its belief about the current context.

We can again represent this process graphically, as shown in Fig 6. In Problems 3.4 and 3.5, you derived an expression for the belief $p_{\text{belief}, t}^A$ at time $t$. Propagate this logic forward in time to fill in the missing elements in Fig 6. The dotted arrows indicate possible dependencies between these quantities; darken the subset of arrows that represent the dependencies we used in Problems 3.4 and 3.5.

![Figure 6: Updating beliefs.](image)

3.7 Think back to Section 2. Which part of the inference process would have to change if the bee does not have direct access to the incoming frequency values, but rather only has access to the output of the detector? How could the bee use its belief about the current context to adapt its color detector?

For the following problems, assume that the bee has direct access to incoming frequency values (i.e., you do not need to incorporate the encoding step mentioned in Problem 3.7). Solve Problem 3.8 analytically, and Problems 3.9-3.10 numerically.

*3.8 How certain is the bee that it is in context A? Write down an expression for the entropy of the posterior, which quantifies this uncertainty. Under what posterior beliefs is the entropy of the posterior distribution maximized?

*3.9 Assume that the bee reliably moves between the two contexts every 10 flower encounters. Within a given context, assume that flower colors are distributed randomly (i.e., assume that flower colors are sampled randomly from the Gaussian distributions shown in Fig 1B). Simulate this inference process for many context switches, and plot the posterior belief (averaged over switches) as a function of encounters. Should we expect the bee to be faster at inferring one context versus the other?

*3.10 What would change if the distribution of flowers in each context had the same mean, but different variances?
4 Selecting Actions

In the previous section, we considered how the bee could infer which context it was in (forest or field) based on sensory input. We’re now going to consider how the bee could use this information to guide foraging behavior. For the purposes of this section, we’ll assume that the forests and fields are densely covered with flowers, so that we can approximate flower color as a continuous variable (as a consequence, we will no longer think about time in units of flower encounters, as we did in the previous section). We’re also going to assume that the bee’s goal is to collect as much nectar as possible over some fixed time of foraging, and we’ll also assume that flower color is the only feature of the environment that predicts nectar content.

To solve this problem, the bee needs to learn which flowers contain nectar in which contexts. The tricky part is that the bee has to learn this through experience, by sampling different flowers. How it moves through the environment will determine how it samples, which in turn will determine what it learns about the environment.

We’ll start by making some simplifying assumptions about how the bee can move around in the environment. In reinforcement learning (RL) terms, we do this by constructing what’s called a policy. The policy is often written as $\pi(a|s)$, and it specifies which actions $a$ an animal (or agent) will take, depending on some state $s$ of the environment. We’ll construct a simplified policy for our bee. We’ll assume that the bee flies in straight segments, punctuated by reorientations where it randomly changes directions. We’ll also assume that the bee can control the probability of reorienting depending on its visual input (what color it sees). In this setup, the states are specified by the variable $f$ (the flowers color, in units of frequency; a continuous variable). The actions are changes in direction, and they come in two types: “fly straight”, or “randomly reorient”.

In a perfect and all-knowing scenario, the bee would never reorient when it is moving toward nectar, and it would always reorient when it is moving away from nectar. However, in practice, the bee doesn’t know with certainty whether it’s moving toward or away from nectar. Rather, it has to estimate this based on its own experience. We’re going to introduce the notion of value in order to build this estimate. The value is often written as $v_\pi(s)$, and it is an estimate of the long-term reward, or return, that an agent expects to receive from beginning in a particular state $s$ and following a particular policy $\pi$ thereafter. In other words, it measures how “good” it is to be in a particular state, where “good” is defined in terms of return. As we’ll come to later, RL algorithms are designed to maximize return. (For more background, see [6])

4.1 To begin, let’s assume that a neuron in the bee’s brain constructs a linear estimate of value, which we’ll denote $\hat{v}$, based on flower color:

$$\hat{v}(f) = w_1 f + w_0$$

(1)

This function describes the bee’s expectations about a given sensory input, or frequency $f$; larger values of $\hat{v}$ correspond to frequencies that are expected to be more rewarding in the long run (note that if we want to know the expectation at a particular time $t$, we would evaluate this expression using the frequency observed at time $t$, i.e. $\hat{v}(f(t))$). In an ideal scenario (i.e., one in which the bee has already learned about the environment), what would you expect the sign of $w_1$ to be if the bee is in the forest? In the field?

4.2 Equation 1 performs a simple form of function approximation; instead of a “lookup table” that independently specifies the estimated value for every possible flower color, it expresses the relationship between value and flower color using a continuous function that is parameterized by $w_1$ and $w_0$. How could you build a more flexible approximation for $\hat{v}$? Is there any downside of doing this?

4.3 We’re now going to use the value function in Equation 1 to guide the bee’s policy. Let’s assume that the probability of reorienting at any given time $t$ depends on whether the sensory input at time $t$ is expected to be better or worse than the previous sensory input at time $t - 1$. What quantity should be on the horizontal axis of Fig 7A? Can you think of other systems that use a policy like this?

4.4 The function in Fig 7A can be parameterized as $P(x) = 1/(1 + \exp(mx + b))$. What does the slope $m$ of this function signify? What would a more deterministic policy look like? Draw it in Fig 7A. Use this policy (including the distribution of changes in orientation shown in Fig 7B) to sketch a few example trajectories on top of the heatmaps shown in Fig 7C-D, assuming that the bee starts at the position indicated by the ’$c$’, initially selects one of the three actions indicated by arrows, and moves the length of an arrow with each timestep. Assume that the bee maintains the value functions shown in panels Fig 7C-D.
Now let’s think about how the bee might modify its behavior based on its experiences. After landing on a flower with color $f$, let’s assume the bee received a reward $r$ equal to the amount of nectar it could collect from the flower. The bee can now use this information to update its estimate $\hat{v}$ by adjusting the parameters $w_1$ and $w_0$. For simplicity, we’ll assume that the bee only adjusts the parameter $w_1$, and keeps the value of $w_0$ fixed. In principle, there are many possible ways to modify this parameter; RL algorithms provide a principled choice based on maximizing return. A simple yet powerful method for doing this in an on-line fashion (i.e., during behavior) is called “temporal difference” learning, and it takes the following form:

$$\Delta w_1 = \alpha \delta(t) \frac{d\hat{v}}{dw_1} \bigg|_{f(t-1)}$$

(2)

This equation computes the derivative of the value function with respect to the parameter $w_1$, and then evaluates this expression using the frequency observed at time $t-1$. In this expression, $\alpha$ is a learning rate, and $\delta(t)$ is the so-called “reward prediction error”:

$$\delta(t) = r(t) + \gamma \hat{v}(f(t)) - \hat{v}(f(t-1))$$

(3)

Here, $\hat{v}(f(t-1))$ is the expected value based on the visual input at time $t-1$, and $r(t)$ is the actual reward received at time $t$. The additional term $\gamma \hat{v}(f(t))$ accounts for “discounted future rewards”, where $\gamma$ is referred to as a “discount factor”. If $\gamma = 0$, the agent is purely “myopic” and cares only about current reward; if $\gamma > 0$, the agent cares not only about the current reward, but also about the possibility of accumulating more reward in the future. The discount factor specifies how much the agent weights the possibility of future rewards against the current reward.

Use Equations 1-3 to write out the update equation for $w_1$.

*Note that Equation 2 generalizes to more complex forms of function approximation in which the value function depends on a vector of parameters $\vec{w}$. In that case, the weights are updated based on the gradient
of the value function, $\nabla \hat{w} \hat{v}$, evaluated using the frequency at time $t-1$:

$$\Delta \hat{w} = \alpha \hat{\delta}(t) \nabla \hat{w} \hat{v} \bigg|_{f(t-1)}$$  

(4)

4.6 Finally, consider two possible moves that the bee could take from time $t-1$ to time $t$, as shown in Fig 8A. Pick either context (forest or field), and use panels B-C to mark the quantities that you would need to compute the updates that you wrote in the last problem. You can assume that frequencies, values, and rewards are all positive quantities.

Next, in panel D, mark the probability that the bee will reorient at time $t$ (thereby guiding how it moves from time $t$ to time $t+1$), and update the value function in panel E (here, you can assume that the lowest frequency is zero, so that any updates to the value function will share the same y-intercept as the dotted lines in panel E). Assume here that $\gamma = 0$. How would this update differ if you chose the other context? Would the update be larger or smaller if $\gamma = 1$?

4.7 Think back to Sections 2 and 3. What happens if the bee suddenly changes context (i.e., moves from the forest into the field)? Discuss how you might change this learning process in light of the previous sections on sensory coding and inference.

*4.8 If you’ve made it this far, try coding this up and simulating the behavior of your model bee. First, make a list of the ingredients you’d need to specify in order to simulate the learning process.

This list should include definitions for each of the functions in Fig 7, along with initial conditions. You’ll want to define a frequency landscape for your model bee to navigate; one possibility is to use a superposition of Gaussians that are scaled and translated with respect to one another. The MATLAB function ‘peaks’ implements the following function, which you can modify to create your own landscape:

$$f(x, y) = 3(1-x)^2 \exp(-(x^2) - (y+1)^2) - 10(x/5 - x^3 - y^5) \exp(-x^2 - y^2) - \frac{1}{3} \exp(-(x+1)^2 - y^2)$$
Suggested reading


