Notes on perception2 k-snipe learning algorithm in free-agent

Marshall Abrams January 23, 2017

In `free-agent.perception2`, the function `k-snipe-pref` does this (from the docstring):

> Decides whether snipe eats mush, and updates the snipe's mush-pref in response to the experience if so, returning a possibly updated snipe along with a boolean indicating whether snipe is eating. (Note that the energy transfer resulting from eating will occur elsewhere, in response to the boolean returned here.)

More specifically, `k-snipe-pref` decides to eat iff the sign of the (scalar) sensory input is such that size sensory input — midpoint between actual mushroom sizes has the same as the sign of the snipe's (noisy) mushroom preference `mush-pref`. (The midpoint is thus treated as an innate parameter, or one that was learned earlier, as is the distance between the two mushroom sizes below.) The idea is that a positive value for `mush-pref` means "I prefer large mushrooms," while a negative value means "I prefer small mushrooms." (That doesn't mean that an individual with a positive preference will only eat large mushrooms, however, since the sensory data is quite noisy.)

If the snipe does eat, this allows it to collect information about whether the mushroom is nutritious or somewhat poisonous. (This is similar to what rats do, as I understand it, refusing to eat a kind of food if eating it was followed by illness.) When a k-snipe eats, `k-snipe-pref` calculates a new value for its `mush-pref` by calling `calc-k-pref`:

> Calculate a new mush-pref for a k-snipe. Calculates an incremental change in mush-pref, and then adds the increment to mush-pref. The core idea of the increment calculation is that if the (somewhat random) appearance of a mushroom has a larger (smaller) value than the midpoint between the two actual mushroom sizes, and the mushroom's nutrition turns out to be positive, then that's a reason to think that larger (smaller) mushrooms are nutritious, and the opposite for negative nutritional values. Thus it makes sense to calculate the increment as a scaled value of the product of the mushroom's nutrition and the difference between the appearance and the midpoint. Thus positive values of mush-pref mean that in the past large mushrooms have often be nutritious, while negative values mean that small mushrooms have more often been nutritious, on average.

Specifically, `calc-k-pref` adds the following increment to the current value of `mush-pref`:

$$\text{nutrition value of mushroom} \times \frac{\text{sensory input} - \text{mushroom midpoint}}{\text{distance from low to high mushroom size}} \times \text{dt}$$

where `dt` is a small number, such as `0.001`. (The numerator of the division above is the reciprocal of the `mush-size-scale` global parameter.) The result of adding the above

increment to `mush-pref` is then returned to `k-snipe-pref`. As explained in the indented docstring above, the general idea is that a positive nutritional value pushes `mush-pref` in the direction of the sensory data's difference from the midpoint.

This algorithm is very roughly like a very simplified version of part of the the algorithm that Feldman and Friston (2010, p. 9) use to model the Posner experimental setup[1] A series of eating experiences here takes on a role like the cue signal in the Posner setup. In F&F's model, the result of the signal is due to an interated calculation in response to it. Here it's not a persistent signal to which the system responds, but a series of size-and-nutrition signals from different mushrooms. In F&F's model, the signal from the cue (along with signals from two other regions) incrementally updates a hidden (vector-valued) variable x. This variable indirectly affects the perception of signals in the other regions by affecting the precision which will scale the effects of those signals. In the perception2 mushroom preference algorithm, there's no direct analog of precision, since `mush-pref`, the closest analog of F&F's x, is simply multiplied by the adjusted input signal, with the result tested for a positive value, in order to decide whether to eat. Scaling the size signal by a precision at this point wouldn't have any effect on the sign of the result, so there's no reason, in this kind of scheme, to do that. Part of the reason that the method can be so simple is just that we're only trying to determine a Boolean value based on a very simple sensory signal. We're not even trying to determine a scalar value from a continuous range.

References

Feldman, H. and K. Friston (2010). Attention, uncertainty, and free-energy. *Frontiers in Human Neuroscience 4*, 215.

[1] F&F's model was in fact my starting point in developing this simple method of learning mushroom preferences, even though what I ended up with is very different and much, much simpler.