# HW1 Computational Neurosceince

**Sepehr Saeedpour**

## 1 Static action choice and reward

### (a)

The sum of probabilities is:

$$p(c=1)+p(c=2)=\frac{\exp(\beta m_1)}{\exp(\beta m_1)+\exp(\beta m_2)}+\frac{\exp(\beta m_2)}{\exp(\beta m_1)+\exp(\beta m_2)} \tag{1}$$

$$=\frac{\exp(\beta m_1)+\exp(\beta m_2)}{\exp(\beta m_1)+\exp(\beta m_2)}=1. \tag{2}$$

Thus, $\sum_{c=1}^{2} p(c)=1$.

### (b)

Divide numerator and denominator of $p(c=1)$ by $\exp(\beta m_1)$:

$$p(c=1)=\frac{1}{1+\exp(\beta(m_2-m_1))}. \tag{3}$$

As $(m_2-m_1)\to+\infty$, $p(c=1)\to 0$. As $(m_2-m_1)\to-\infty$, $p(c=1)\to 1$.

### (c) & (d)

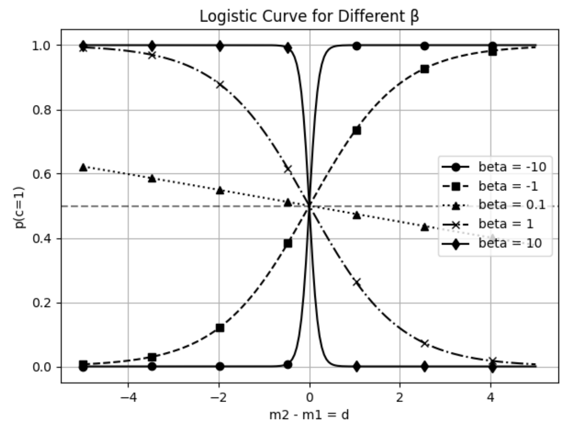

Figure 1: Probability of choosing blue flower as a function of reward difference $d=m_2-m_1$.

For $d = m_2 - m_1$ and $\beta = 1$, $p(c = 1)$ forms a sigmoid curve centered at $d = 0$.

- **Large** $d > 0$**:** $p(c = 1) \to 0$ (bee avoids blue flowers).
- **Large** $d < 0$**:** $p(c = 1) \to 1$ (bee strongly prefers blue flowers).
- $d \approx 0$**:** $p(c = 1) \approx 0.5$ (balanced exploration).

The bee exploits clear reward differences but explores when rewards are similar.

**Parameter** $\beta$ **(see Fig 1):**

- $\beta \to \infty$: Choices become deterministic (pure exploitation).
- $\beta \to 0$: Uniform probabilities (pure exploration).
- $\beta < 0$: Inverts preferences (illogical; standard $\beta > 0$).

**Tradeoff:** High $\beta$ emphasizes exploitation; low $\beta$ promotes exploration.

## (e)

For $N$ flowers, generalize the softmax policy:

$$p(c = i) = \frac{\exp(\beta m_i)}{\sum_{j=1}^{N} \exp(\beta m_j)}. \tag{4}$$

Adjust $\beta$ to balance exploration (low $\beta$) and exploitation (high $\beta$). Optimal learning often requires starting with lower $\beta$ values to encourage exploration and gradually increasing it to focus on exploitation

## (f)

,

When the rewards $r_i(t)$ of the $N$ flowers change over time, the bee needs to continuously update its internal estimates $m_i(t)$ to track these changes. One of the approaches for adapting the estimates and update internal estimates dynamically is EMA(exponential moving average):

$$m_i(t + 1) = (1 - \alpha) \cdot m_i(t) + \alpha \cdot r_i(t + 1) = m_i(t) + \alpha \left[r_i(t + 1) - m_i(t)\right], \tag{5}$$

where $\alpha \in (0, 1)$ is the learning rate. This creates a weighted average that gives more importance to recent rewards and gradually "forgets" older observations.

## (g)

If $r_i(t) = r_i$ (constant) for all $t$, we can solve this recurrence relation:

$$\begin{aligned} m_i(t) &= (1-\alpha)\cdot m_i(t-1) + \alpha \cdot r_i && (6)\\ &= (1-\alpha)\cdot[(1-\alpha)\cdot m_i(t-2) + \alpha\cdot r_i] + \alpha\cdot r_i && (7)\\ &= (1-\alpha)^2\cdot m_i(t-2) + \alpha\cdot r_i\cdot[1+(1-\alpha)] && (8)\end{aligned}$$

Continuing this expansion, we get:

$$\begin{aligned} m_i(t) &= (1-\alpha)^t\cdot m_i(0) + \alpha\cdot r_i\cdot\sum_{k=0}^{t-1}(1-\alpha)^k && (9)\\ &= (1-\alpha)^t\cdot m_i(0) + r_i\cdot(1-(1-\alpha)^t) && (10)\end{aligned}$$

As $t\to\infty$, $(1-\alpha)^t\to 0$ (since $0<\alpha<1$), therefore:

$$\lim_{t\to\infty} m_i(t) = r_i \tag{11}$$

The time constant $\tau$ for the EMA update rule is:

$$\tau = \frac{1}{\alpha} \tag{12}$$

- High $\alpha$: Faster convergence but more susceptible to noise
- Low $\alpha$: Slower convergence but more stable estimates

**Dependence on $\beta$:**

The parameter $\beta$ does not directly appear in the update equation for $m_i(t)$, but it affects the frequency at which the bee visits each flower through the softmax policy:

$$p(c=i) = \frac{\exp(\beta m_i)}{\sum_{j=1}^{N}\exp(\beta m_j)} \tag{13}$$

The effect of $\beta$ on convergence:

- High $\beta$: The bee mostly visits flowers with the highest estimated rewards. In fact, fast convergence for high-reward flowers but potentially slow or no convergence for others
- Low $\beta$: The bee distributes visits more uniformly across all flowers. More uniform convergence across all flowers but at a slower overall rate.

## 2 Temporal-difference learning with discounting

The value function $V(s_t)$ can be rewritten recursively as:

$$\begin{aligned} V(s_t) &= r(s_t) + \gamma \left[ r(s_{t+1}) + \gamma r(s_{t+2}) + \gamma^2 r(s_{t+3}) + \dots \right] && (14) \\ &= r(s_t) + \gamma V(s_{t+1}) && (15) \end{aligned}$$

This recursive relationship is known as the Bellman equation. During learning, our estimate of $V(s_t)$ may not satisfy this equation exactly. The temporal-difference error represents the difference between the right-hand side and left-hand side of this equation:

$$\delta_t = r(s_t) + \gamma V(s_{t+1}) - V(s_t) \tag{16}$$

To update our estimate of the value function, we adjust it in the direction of the TD error, scaled by a learning rate $\epsilon$:

$$V(s_t) \to V(s_t) + \epsilon \left( r(s_t) + \gamma V(s_{t+1}) - V(s_t) \right) \tag{17}$$

This is the temporal-difference learning rule given in equation (2). This update allows the agent to learn online from incomplete sequences, using only the immediate reward and the current estimate of the next state's value, without waiting for the final outcome.

## 3 Models for the value function

### (a)

To find the parameters $\mathbf{w} = (w_1, w_2)$ that the agent has learned:

For the first state $(1, 0)$:

$$V((1,0)) = w_1 \cdot 1 + w_2 \cdot 0 = w_1 = \alpha \tag{18}$$

For the second state $(0, 1)$:

$$V((0,1)) = w_1 \cdot 0 + w_2 \cdot 1 = w_2 = \beta \tag{19}$$

Therefore, $\mathbf{w} = (w_1, w_2) = (\alpha, \beta)$

Now, to find the value of a state where both stimuli are present $(1, 1)$:

$$V((1,1)) = w_1 \cdot 1 + w_2 \cdot 1 = \alpha + \beta \tag{20}$$

This type of generalization is a linear combination of the values of individual features. It assumes that the contributions of different stimuli to the overall value are additive and independent.

1. **Scenario where this generalization makes sense:**
   In a restaurant recommendation system, if a user likes restaurants with outdoor seating (value $\alpha$) and also likes restaurants with vegetarian options (value $\beta$), then a restaurant with both features would be valued at approximately $\alpha + \beta$. The features contribute independently to the overall preference.

2. **Scenario where this generalization does not make sense:**
   In drug interactions, if medication A produces a certain beneficial effect (value $\alpha$) and medication B produces another beneficial effect (value $\beta$), their combination might produce dangerous side effects or reduced efficacy due to chemical interactions. The actual value of combining them could be much lower than $\alpha + \beta$ or even negative, as the effects are not independent.

## (b)

To derive the temporal-difference (TD) learning rule for the parameters $\mathbf{w}$ when the value function is $V(\mathbf{u}) = \mathbf{w} \cdot \mathbf{u}$, we'll start with defining a suitable loss function and then derive the learning rule.

1. Define the loss function as the squared TD error:

$$\begin{aligned} L(\mathbf{w}) &= \frac{1}{2}\delta_t^2 && (21) \\ &= \frac{1}{2}(r(s_t) + \gamma V(s_{t+1}) - V(s_t))^2 && (22) \\ &= \frac{1}{2}(r(s_t) + \gamma \mathbf{w} \cdot \mathbf{u}_{t+1} - \mathbf{w} \cdot \mathbf{u}_t)^2 && (23) \end{aligned}$$

   where $\delta_t$ is the TD error.

2. To minimize this loss function, we use gradient descent. The update rule is:

$$\mathbf{w} \to \mathbf{w} - \epsilon \nabla_{\mathbf{w}} L(\mathbf{w}) \tag{24}$$

3. Calculate the gradient of the loss function with respect to $\mathbf{w}$:

$$\begin{aligned} \nabla_{\mathbf{w}} L(\mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2}(r(s_t) + \gamma \mathbf{w} \cdot \mathbf{u}_{t+1} - \mathbf{w} \cdot \mathbf{u}_t)^2\right] && (25) \\ &= (r(s_t) + \gamma \mathbf{w} \cdot \mathbf{u}_{t+1} - \mathbf{w} \cdot \mathbf{u}_t) \cdot \nabla_{\mathbf{w}}(r(s_t) + \gamma \mathbf{w} \cdot \mathbf{u}_{t+1} - \mathbf{w} \cdot \mathbf{u}_t) && (26) \\ &= \delta_t \cdot (\gamma \mathbf{u}_{t+1} - \mathbf{u}_t) && (27) \\ &= \delta_t \gamma \mathbf{u}_{t+1} - \delta_t \mathbf{u}_t && (28) \end{aligned}$$

4. Substitute this gradient into the update rule:

$$\begin{aligned} \mathbf{w} &\to \mathbf{w} - \epsilon(\delta_t \gamma \mathbf{u}_{t+1} - \delta_t \mathbf{u}_t) && (29) \\ &\to \mathbf{w} - \epsilon \delta_t \gamma \mathbf{u}_{t+1} + \epsilon \delta_t \mathbf{u}_t && (30) \\ &\to \mathbf{w} + \epsilon \delta_t \mathbf{u}_t - \epsilon \delta_t \gamma \mathbf{u}_{t+1} && (31) \end{aligned}$$

5. This is slightly different from the standard TD learning rule. However, we can adopt a semi-gradient approach, where we ignore the dependency of the target on the current weights. This means treating $\gamma \mathbf{w} \cdot \mathbf{u}_{t+1}$ as a constant with respect to $\mathbf{w}$ when computing the gradient. With this approach:

$$\begin{aligned} \nabla_{\mathbf{w}} L(\mathbf{w}) &\approx (r(s_t) + \gamma \mathbf{w} \cdot \mathbf{u}_{t+1} - \mathbf{w} \cdot \mathbf{u}_t) \cdot \nabla_{\mathbf{w}}(-\mathbf{w} \cdot \mathbf{u}_t) \quad (32) \\ &= -\delta_t \mathbf{u}_t \quad (33) \end{aligned}$$

6. The semi-gradient TD update rule becomes:

$$\begin{aligned} \mathbf{w} &\to \mathbf{w} - \epsilon \cdot (-\delta_t \mathbf{u}_t) \quad (34) \\ &\to \mathbf{w} + \epsilon \delta_t \mathbf{u}_t \quad (35) \end{aligned}$$

Thus, the TD learning rule for the linear value function is:

$$\mathbf{w} \to \mathbf{w} + \epsilon \delta_t \mathbf{u}_t \quad (36)$$

where $\delta_t = r(s_t) + \gamma \mathbf{w} \cdot \mathbf{u}_{t+1} - \mathbf{w} \cdot \mathbf{u}_t$

For a nonlinear value function $V(\mathbf{u}) = f(\mathbf{w} \cdot \mathbf{u})$, applying the semi-gradient approach:

$$\begin{aligned} \delta_t &= r(s_t) + \gamma f(\mathbf{w} \cdot \mathbf{u}_{t+1}) - f(\mathbf{w} \cdot \mathbf{u}_t) \quad (37) \\ \nabla_{\mathbf{w}} L(\mathbf{w}) &\approx -\delta_t \nabla_{\mathbf{w}} f(\mathbf{w} \cdot \mathbf{u}_t) \quad (38) \\ &= -\delta_t f'(\mathbf{w} \cdot \mathbf{u}_t) \mathbf{u}_t \quad (39) \end{aligned}$$

Therefore, the TD learning rule for the nonlinear case becomes:

$$\begin{aligned} \mathbf{w} &\to \mathbf{w} - \epsilon \cdot (-\delta_t f'(\mathbf{w} \cdot \mathbf{u}_t) \mathbf{u}_t) \quad (40) \\ &\to \mathbf{w} + \epsilon \delta_t f'(\mathbf{w} \cdot \mathbf{u}_t) \mathbf{u}_t \quad (41) \end{aligned}$$