INFORMATION PROCESSING AND THE BRAIN
2019/2020
ADDITIONAL EXERCISES

RUI PONTE COSTA

1. Visual cortex and CNNs

(1) Simple cells, are commonly found in the visual cortex. Explain their key properties?

Answer: Simple cells are neurons in the visual cortex discovered by Hubel and Wisel. These neurons responde to particular visual features, namely lines of particular orientations

(2) Give the Gabor filter model of simple cells and explain its different components. Also, explain how this model can be used to model complex cells and to model a neuron firing rate.

Answer: Simple cells can be modelled as $v = f(wu)$ where u is the visual input, w are the weights or filters, f is the activation function, typically a non-linear transfer function, and v models the firing rate of the simple cells. The particular behaviour of simple cells can be modelled using Gabor filters. Gabor filters are defined as:

$$w \sim g(x, y; \lambda, \theta, \phi, \gamma) = \exp\left(- \frac{x'^2 + \gamma^2 y'^2}{2\sigma^2} \right) \cos\left(2\pi \frac{x'}{\lambda} + \phi \right)$$

with $y' = -x \sin\theta + y \cos\theta$ and $x' = x \cos\theta + y \sin\theta$. Gabor filters are the product of a Gaussian and a sinusoid term, and the parameter θ defines the angle of the filter, which captures the orientation of the simple cells as observed in the visual cortex. The remaining parameters (σ , ϕ , λ and γ) jointly define the overall shape of the filter.

Complex cells, responde to the same orientation at different input locations. A simple model of complex cells is to combine multiple simple cells as $v = f(wg)$, where g is given by the Gabor model defined above. A more biologically realistic model would be to use the firing rate model of simple cells defined above $g = f(wu)$.

(3) Explain how these different elements inspired convolutional neural networks.

Answer: There is a strong parallelism between convolutional neural networks (CNNs) and the visual cortex. First, simple cells respond to specific features in the input (e.g. bars of a particular orientation). In CNNs the filters w are typically learned through gradient descent using the backpropagation algorithm. The filters learned this way resemble the ones observed in the visual cortex. The behaviour of complex cells is akin to the convolution operation in CNNs, as a given filter is applied across multiple input locations. The neuronal activation functions used by real neurons are similar to the thresholding functions used in CNNs. Finally, pooling can be implemented in the brain via specific connectivity between brain areas and layers in CNNs correspond to different brain areas.

2. Sparse coding

(1) Briefly explain what sparse coding is?

Answer: In sparse coding is a neuronal coding framework in which input items are represented by a small portion of strongly active neurons. In addition, different stimuli active a different group of neurons.

(2) What is the cost function typically used in sparse coding? Explain its different elements and how these lead to a sparse representation.

Answer: The cost function typically used in sparse coding is defined as follows

$$\text{argmin}_{W,V} = \|U - WV\|_2^2 + \lambda\|V\|_1$$

where U is the input, W the features or dictionary and V the representation. V can be interpreted as the firing rates of postsynaptic neurons. The first term ensures that information is preserved using a L2-norm, whereas the second term ensures that a sparse representation is obtained after optimization using a L1-norm. λ defines the trade-off between these two terms.

(3) What is the weight learning rule used by Olshausen and Field Nature 1996? Briefly explain how it relates to biology.

Answer: The learning rule used by Olshausen and Field Nature 1996 is $\Delta w \propto \underbrace{UV}_{\text{Hebbian term}} - \underbrace{WV^2}_{\text{postsynaptic term}}$. The first term UV of the learning rule follows Hebbian principles (i.e. cells that fire together wire together), as when only both U and V neurons are active (i.e. their firing rates are higher than 0) it leads to a change in the synaptic weight. The second term WV^2 depends on the postsynaptic activity alone V (i.e. is non-hebbian), and acts as a normalizing (or regularizing) force by decreasing the weights. There is evidence for both terms in biology.

3. Reinforcement learning and Q-learning

(1) Give the equation for the reward prediction as defined in TD-learning and explain how it is related to neuromodulation in the brain.

Answer: The equation for reward prediction error is given by $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$. Dopamine is the neuromodulator believed to encode reward prediction errors, such that when there is a positive error, no error or negative error it has been observed experimentally an increase, no change and decrease in firing rates of dopaminergic neurons as shown by Schultz et al. (1997).

(2) Give the value update function used in Q-learning and briefly describe the different terms.

Answer: The Q-learning equation is given by

$$\underbrace{Q(S_t, A_t)}_{\text{value}} = Q(S_t, A_t) + (\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \gamma \max_a \underbrace{Q(S_{t+1}, a)}_{\text{future value}} - Q(S_t, A_t)}^{\delta\text{, prediction error}})$$

where, Q is the value function for a state S_t action pair A_t, R is the reward and γ is the discount factor, and the max operation selects the action with the maximal Q value.

(3) Given the following reward sequence R and state space sequence S compute the value function for $Q(S_{t=1}, A_{t=1})$, $Q(S_{t=2}, A_{t=2})$ and $Q(S_{t=3}, A_{t=3})$, for the *first*, *second* and *third* update steps. Assume that the value function starts at 0 before any update, $\gamma = 0.9$ and that there is only one action available. Start from the equation given in the previous question.

$S = \{S_{t=1}, S_{t=2}, S_{t=3}\}$
$R = \{R_{t=1} = 0, R_{t=2} = 0, R_{t=3} = 0, R_{t=4} = 1\}$

Answer:
first update:

$$Q(S_{t=1}, A_{t=1}) = 0 + (0 + 0.9 * 0 - 0)) = 0$$

$$Q(S_{t=2}, A_{t=2}) = 0 + (0 + 0.9 * 0 - 0)) = 0$$

$$Q(S_{t=3}, A_{t=3}) = 0 + (1 + 0.9 * 0 - 0)) = 1$$

Second update:

$Q(S_{t=1}, A_{t=1}) = 0 + (0 + 0.9 * 0 - 0)) = 0$

$Q(S_{t=2}, A_{t=2}) = 0 + (0 + 0.9 * 1 - 0)) = 0.9$

$Q(S_{t=3}, A_{t=3}) = 1 + (1 + 0.9 * 0 - 1)) = 1$

Third update:

$Q(S_{t=1}, A_{t=1}) = 0 + (0 + 0.9 * 0.9 - 0)) = 0.81$

$Q(S_{t=2}, A_{t=2}) = 0.9 + (0 + 0.9 * 1 - 0.9)) = 0.9$

$Q(S_{t=3}, A_{t=3}) = 1 + (1 + 0.9 * 0 - 1)) = 1$

Note that because here we assume to only have one possible action, the term $\max_a Q(S_{t+1}, a)$ simplifies to $Q(S_{t+1}, A_{t+1})$.