Coursework for
4G3 Computational Neuroscience
Assignment 1

Máté Lengyel

assigned: February 14, 2025
due: February 28, 2025, 4pm
submit through Moodle
don't forget to attach the correct cover page

In this handout,

- text in blue corresponds to things you have to do / implement. To perform your simulations, you may use any programming language of your choice.
- text in orange corresponds to questions you need to answer in some form in your report.

Please write up your findings in a report to be submitted on Moodle in PDF format, and *include all your code in the Appendix*. Please do not forget to include the correct cover page (which depends on your course, see the cover page section on Moodle) and clearly include either your candidate number or your name on it as required by the form.

Your report should address all the questions raised in this handout, be structured around the sections herein, and ****be a maximum of eight A4 pages**** excluding any Appendix (minimum font size 11pt, minimum margins 1.5cm on each side, and you are encouraged to make your submission shorter than this maximum as long as you solve all exercises). You are very much encouraged to think of data/results visualisations to best support the exposition and interpretation of your results.

Please make sure to include a statement at the beginning of your report declaring whether you used generative AI tools (e.g. ChatGPT) for your simulations and / or writing your report. If you did, prepare an Appendix including all prompts you entered while using these tools in connection with this Assignment (including for literature search, explaining scientific concepts, help with code writing, help with writing / polishing the text or the figures of the report). As a word of caution, use these tools with great care: you will be responsible for any result you present and, if asked, you will need to be able to give evidence that you understand how exactly those results were obtained.

Question I. Reinforcement learning **[15 points]**

In this exercise you will study simple generalisations of the temporal-difference (TD) learning algorithm we discussed in lectures.

The form of the TD algorithm presented in the lectures builds up an estimate of the value for each separate state, thus leading to a "lookup table" representation. This is infeasible when the state space is too large, for two reasons. First, it might require us to store too many values. Second, and more fundamentally, typically several visits to the same state are needed to build a reliable estimate of its value, but we may never enter the exact same state twice (or enough times). This could be circumvented if knowledge about the value of one state could inform estimates for other states – i.e. if we had a representation that allowed *generalisation* across states.

A classical way to achieve generalisation, which has recently become very popular in cutting-edge machine learning applications, is based on the idea of *function approximation*. That is, instead of explicitly representing the (approximate) value function, $\hat{V}^{\pi}(s)$, for each state, s, and estimating these values directly, we use some parametric function to approximate the value function, $\hat{V}^{\pi}(s) = f(s; \mathbf{w})$, and iteratively estimate its parameters, $\mathbf{w}$ (we use boldface to denote vectors), using the same bootstrapping idea underlying lookup table-TD:

$$w_k \leftarrow w_k + \epsilon \left[r_t + \gamma \hat{V}^{\pi}(s_{t+1}) - \hat{V}^{\pi}(s_t) \right] \frac{\partial f(s_t, \mathbf{w})}{\partial w_k} \tag{1}$$

where t indexes time, r is immediate reward, and γ is the discounting factor – as in the lecture handouts.

1. Show that Eq. 1 reduces to the lookup table-form of TD discussed during lectures with the appropriate choice of $f(s; \mathbf{w})$. What is the form of $f(s; \mathbf{w})$ in this case, and what are its parameters?

A special case of function approximation is *linear function approximation*, when the function approximator is written as a weighted sum of the output of "feature detectors", $\phi_k(s)$, each computing some simple and fixed function of s (detecting some feature in it), and its parameters are the weighting factors controlling the relative contribution of each of these features to the value function estimate: $f(s, \mathbf{w}) = \sum_k w_k \, \phi_k(s)$.

2. What is the form of the update shown in Equation 1 in this special case?

In the lectures, the way we mapped the abstract notion of a "state" in TD (and, generally, in Markov decision processes) to a classical conditioning experiment, was that we assumed that each state corresponds to one time step in a trial of the experiment. However, the (often implicit) requirement for TD to work (and in fact, the assumption underlying essentially all reinforcement learning algorithms) is the so-called Markov property of the states on which it operates: that the future of a system should be independent of its past given its current state (and the currently executed action). This poses a problem for our mapping because in a generic classical conditioning experiment what happens in future time steps may not only depend on whatever is happening in the current time step, but also what happened over some history of past time steps (which stimulus, if any, was presented, exactly how long ago, etc). Thus, our original naïve mapping from time steps to states is only appropriate as long as every trial goes through the exact same sequence of events, so essentially when there is only one (conditioned) stimulus, appearing always at the same exact time, and deterministically predicting the same amount of reward with the same delay in every trial.

To handle the more general situation with multiple rewards, potential stochasticity, etc, we need to use a richer state representation, for example one which includes a record of the past history of stimuli as the state. The most straightforward such representation is a stimulus-by-time step matrix, $\mathbf{S}$, such that $S_{i\tau}$ is the extent to which stimulus i was present τ time steps ago, with $0 \leq \tau \leq T_{\text{mem}}$, where T_{mem} is the memory span of the agent. (Note that τ indexes time relative to the current "absolute" time, t, in the reverse direction, so if the presence of stimulus i as a function of time is given by $y_i(t)$, then $S_{i\tau} = y_i(t - \tau)$). $\mathbf{S}$ now spans a potentially very large state space which motivates the use of function approximation.

3. A *"tapped delay line" representation* uses feature detectors that are indexed by both stimulus and time such that the feature detector index introduced above is the tuple $k = \{i, \tau\}$, and $\phi_{i\tau}(\mathbf{S}) = S_{i\tau}$, i.e. each feature detector simply returns the level of one stimulus at a specified time ago. Using this representation, and the update rule derived in Question I.2, implement the TD algorithm. Simulate $N = 201$ trials of an experiment with (within-trial) time going from $t = 0$ to 25 s (in steps of $\Delta t = 0.5$ s), a single stimulus presented at 10 s, such that its level is described by a "spike" $y_i(t) = g(t - 10\text{ s})$, where $g(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$, and reward presented in a temporally smoother way around 20 s, such that its level varies as $r(t) = \frac{1}{2}\,\mathcal{G}(t;\, 20\text{ s},\, 1\text{ s})$ where $\mathcal{G}(x;\, \mu, \sigma) = e^{-\frac{(x-\mu)^2}{2\,\sigma^2}}$ is the Gaussian function with mean μ, standard deviation σ, and maximal value 1. Use $\gamma = 1$ (i.e. no discounting), a learning rate $\epsilon = 0.2$ (for the given time step size), and a memory span $T_{\text{mem}} = 12$ s (so that $\mathbf{S}$ is a 1×25 matrix).

   (a) Plot the stimulus and the reward as a function of time.

   (b) For every 10th trial (starting from the first one, so 21 example trials in total), plot the following key variables as functions of time, t:

      i. the value, $\hat{V}(t)$,

      ii. the temporal difference of the value $\Delta\hat{V}(t) = \gamma\,\hat{V}(t) - \hat{V}(t - \Delta t)$, and

      iii. the temporal difference learning error $\delta(t) = r(t - \Delta t) + \Delta\hat{V}(t)$.

      Preferably, use the format seen in lecture slides: three panels underneath each other for the three variables, sharing the same x- (time) axis. In each panel, you can use lines with different colours corresponding to the 21 example trials that need to be plotted.

   (c) Comment on your results. Why do the time courses of the variables look like they do, and why do these time courses change across learning in these particular ways?

   (d) The general observation in actual data about dopamine cell activation discussed during lectures is that before (or at the beginning of) learning, dopamine cells are active at the time of the reward, while after (or towards the end of) learning they are active at the time of the stimulus. However, this "shift" happens in a way that no dopamine activation is usually seen between these two times: instead the "bump" of activity at the time of the reward gradually disappears while a bump at the time of the stimulus gradually develops. What variable in the TD model is supposed to correspond to dopamine activity? Does its behaviour in your model agree with these observations?

4. Another memory-based representation often used for function approximation is a *"box-car" representation* in which feature-detectors are described as $\phi_{i\tau}(\mathbf{S}) = \sum_{u=0}^{\tau} S_{iu}$, i.e. they detect the total presence of a stimulus over some past history. Using this representation, and a learning rate $\epsilon = 0.01$, simulate the same experiment (with all other parameters being the same) as above.

   (a) Plot the same three variables (in the same trials and in the same format) as before.

   (b) Comment on your results. Did anything change in the first and last trial from the previous results? Did anything change in the intervening trials, during the course of learning, compared with the previous results? If there were changes, explain why they occurred.

   (c) How do these results relate to the experimental observation described above about the "shift" of the dopamine signal over the course of learning?

   (d) Why did we need to change the learning rate? What would have happened if we had left it unchanged?

5. An often used classical conditioning paradigm uses *partial reinforcement* such that only some (randomly chosen) fraction, p, of trials ends with the reward presented, in the rest of the trials the reward is omitted. Simulate $N = 1000$ trials with $p = 0.5$, using the same model as above (with boxcar representation).

   (a) For the last 100 trials, plot the average of each of the same three variables as above, separately for

      i. rewarded trials,

      ii. unrewarded trials, and

      iii. for all trials.

   (b) Which variables show or don't show a difference between these three kinds of averages? Explain your findings.

   (c) Dopamine neurons have a limited and *asymmetric dynamic range*: they can only signal negative quantities (below their baseline) to a limited extent, and they also saturate at high firing rates so they can also only signal large positive quantities to a limited extent. Thus, a simple model of dopamine activity is:

$$\mathrm{DA}(x) = \begin{cases} x/\alpha & \text{if } x < 0 \\ x & \text{if } 0 \leq x < x^* \\ x^* + (x - x^*)/\beta & \text{if } x^* \leq x \end{cases}$$

   where x is the variable that dopamine activity encodes, and $\alpha = \beta = 6$ and $x^* = 0.27$. Plot the time course of the dopamine signal averaged across all the last 100 trials. How and why is it different from the average time course of the variable it encodes?

6. Simulate a set of similar experiments, but each using a different reward probability: $p \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$. Plot the average dopamine time course (in the last 100 trials, as above) for each of these experiments in a single figure (colour coding the corresponding value of p). What do you notice?

7. As a function of p, plot the (average) level of dopamine at its peak around the time of the stimulus, and also at the time of the reward. Describe in words how the level of dopamine depends on p at these two times, and explain these dependencies.

8. There have been suggestions that dopamine might signal uncertainty, instead of a prediction error. What pattern in the dopamine signal you obtained might suggest this to be the case? Do you have an alternative explanation for this pattern?

Question II. Representational Learning **[15 points]**

This exercise is about a simplified version of the sparse coding model covered during the lectures – also known as independent components analysis (ICA). In this variant, the observation noise is ignored, $\sigma_i = 0$, and there are no recurrent connections, $\mathbf{W}^{\text{rec}} = \mathbf{0}$.

The exercise follows a simple but powerful procedure for assessing (and eventually improving) the quality of an internal model (such as that implied by the sparse coding model). A well-calibrated internal model, which describes data well, should have the prior distribution over latent variables $p(x)$ and the (empirical) marginal posterior distribution $\frac{1}{|\text{data set}|}\sum_{y \in \text{data set}} p(x|y)$ be (roughly) equal. (Note that in the context of the sparse coding model each posterior $p(x|y)$ is a delta distribution, i.e. a single value of x, so the marginal posterior can simply be constructed by collecting the outputs the system generates across a set of inputs.) When the prior and the marginal posterior are not equal, that is taken to be an indication that the model needs to be improved. One way to improve it is to define a new model in which a different prior is defined over the latent variables that matches better the marginal posterior of the current model.

Preparations:

Download the file `representational.zip` from the module's Moodle page. This file contains a Matlab data file `representational.mat` and the Matlab functions `plotIm.m`, `checkgrad.m` and `minimize.m`.

The data file contains three variables:

- `Y` is a 32000×1024 array containing 32000 image patches that are each 32 by 32 pixels in size, but which have been reshaped into the array. Elements of `Y` are denoted $\mathrm{y}_{n,d}$ below.
- `R` is a 1024×256 matrix containing the feed-forward weights (corresponding to $\mathbf{W}^{\text{ff}}$ in the lecture slides) from the simplified sparse coding model that has been trained on a different set of natural image patches. Elements of `R` are denoted $\mathrm{r}_{d,k}$ below.
- `W` is a 1024×256 matrix containing the generative weights from the same model. Elements of `W` are denoted $\mathrm{w}_{d,k}$ below.

The Matlab functions will help you write the code and process the results for this section of the assignment. For example, the function `plotIM` lets you visualise a subset of the image patches `plotIm(Y(1:100,:)')` or view the generative weights `plotIm(W)`. You will see that the generative weights define localised and oriented dark-light edges at a range of spatial frequencies. The feed-forward weights can be viewed in an identical manner, but they are less easy to interpret. For this reason, results should be analysed and interpreted in terms of the generative weights where possible.

Exercises:

1. Recall that the goal of the sparse coding model is to recover "components", latent variables, which are sparsely and independently distributed. Investigate the extent to which the model fulfils this objective by computing the values of latent variables from the data using the feed-forward weights, $\mathrm{x}_{n,k} = \sum_d \mathrm{y}_{n,d}\, \mathrm{r}_{d,k}$.

   (i) For latent variables k of your choice, estimate their empirical marginal (posterior) probability distribution by forming a histogram from the values $\{x_{n,k}\}_{n=1}^N$ and plotting your estimate of $p(\mathrm{x}_k)$ as a function of x_k. Are the components sparse?[1] Make sure you include components corresponding to generative weights whose spatial structure has high frequency.

   (ii) For pairs of components k_1 and k_2 of your choice, estimate the pairwise joint distribution of the components $p(\mathrm{x}_{k_1}, \mathrm{x}_{k_2})$ by forming a 2D histogram of the values $\{x_{n,k_1}, x_{n,k_2}\}_{n=1}^N$. Renormalise slices of this histogram to form an estimate of the conditional distribution $p(\mathrm{x}_{k_2}|\mathrm{x}_{k_1})$. Plot this conditional using x_{k_1} and x_{k_2} as the x- and y-axes and using colour to indicate $p(\mathrm{x}_{k_2}|\mathrm{x}_{k_1})$. Are the components independent?[2] Make sure you include pairs of components with corresponding generative weights that define edges whose location and orientation is similar.

2. Investigate the form of the residual dependencies using a conditional model which assumes that each coefficient is drawn from a Gaussian distribution with a variance $\sigma_k^2(\mathbf{x}_{\neq k};\theta)$ that depends quadratically on all of the other components,

$$p(\mathrm{x}_k|\mathbf{x}_{\neq k},\theta) = \frac{1}{\sqrt{2\pi\sigma_k^2(\mathbf{x}_{\neq k};\theta)}} \exp\left(-\frac{1}{2\sigma_k^2(\mathbf{x}_{\neq k};\theta)}\mathrm{x}_k^2\right),$$

$$\sigma_k^2(\mathbf{x}_{\neq k};\theta) = \sum_{j\neq k} a_{k,j}\mathrm{x}_j^2 + b_k \quad \text{where} \quad a_{k,j} > 0 \text{ and } b_k > 0.$$

   Here $\mathbf{x}_{\neq k}$ denotes all of the variables except x_k.

   Estimate the parameters $\theta = \{a_{k,j}, b_k\}$ from the components X= $\{\mathrm{x}_{n,k}\}_{n=1,k=1}^{N,K}$ by maximising the conditional log-likelihood of the parameters,

$$\mathcal{L}(X;\theta) = \sum_{n=1}^N \sum_{k=1}^K \log p(\mathrm{x}_{n,k}|\mathbf{x}_{n,\neq k},\theta).$$

   Bear in mind the following pointers:

[1] Hint: a distribution is sparse when it is more "peaked" and has "heavier" tails than a Gaussian with equal mean and variance, i.e. it assigns a higher probability to outcomes that are very close or very from from the mean, but smaller probabilities to outcomes with intermediate values. This is best seen by plotting probabilities on a logarithmic axis. (For reference, what does a Gaussian look like on a logarithmic probability axis?)

[2] Hint: recall that two variables X and Y are independent if (and only if) $p(X|Y) = p(X)$, i.e. if $p(X|Y)$ is the same function of X for any value of Y. What does that mean visually in the context of the plot you are plotting here?

- the function `minimize` can be used to carry out the optimisation;
- `minimize` requires a Matlab function that computes the negative conditional likelihood and the derivatives with respect to the parameters being optimised (for detailed information about usage, see `help minimize`);
- in order to handle the positivity constraints, the parameters should be expressed in terms of logs i.e. $\log(a_{k,j})$ and $\log(b_k)$;
- `minimize` requires the parameters that are being optimised to be passed in as a vector which involves stacking $\log(a_{k,j})$ and $\log(b_k)$;
- consistency between your objective function and gradients can be verified numerically using the `checkgrad` function (for more information about usage, see `help checkgrad`);
- you may only want to use a subset of the data for training, at least initially (e.g. $\approx$ 8000);
- plot the objective function returned by `minimize` to verify convergence.

Your write up should include the derivatives of the objective function and a brief description of your implementation.

3. Once the model has been trained, consider the normalised variables, $\mathrm{c}_k = \mathrm{x}_k/\sigma_k(\mathbf{x}_{\neq k};\theta)$. Plot the marginal distributions $p(\mathrm{c}_k)$ of these variables as you plotted $p(\mathrm{x}_k)$ above, and compare the two kinds of distributions. Verify your conclusions by computing the excess-kurtosis for both variables.[3] Similarly, plot and compare the conditional distributions $p(\mathrm{c}_{k_2}|\mathrm{c}_{k_1})$ to $p(\mathrm{x}_{k_2}|\mathrm{x}_{k_1})$ computed above. What do you notice? Explain your observations.

4. Pick a component k of your choice and plot its generative weight using `plotIm(W(:,k))`. Plot the generative weights of the ten components for which the parameter $a_{k,j}$ associated with that components is largest. Repeat for alternative components. What do you observe and what conclusions can be drawn from this about the statistics of natural images?

5. Figure 1 shows electrophysiological data from primary visual cortex (V1). The mean response rate of a V1 neuron varies as a function of the contrast of an optimally oriented grating presented in the classical receptive field, called the "signal". The tuning curve varies when a masker is present in the surround. Parallel maskers of increasing contrast suppress the response (a) whilst orthogonal maskers have little effect (b).

   By identifying the neural response with one of the variables considered above, provide a computational interpretation of these experimental results.

[3] The excess kurtosis is the kurtosis of a distribution (normalised fourth moment, e.g. as computed by the matlab function `kurtosis` from samples) minus 3 (the kurtosis of the normal distribution). As such, it is a measure of sparseness.

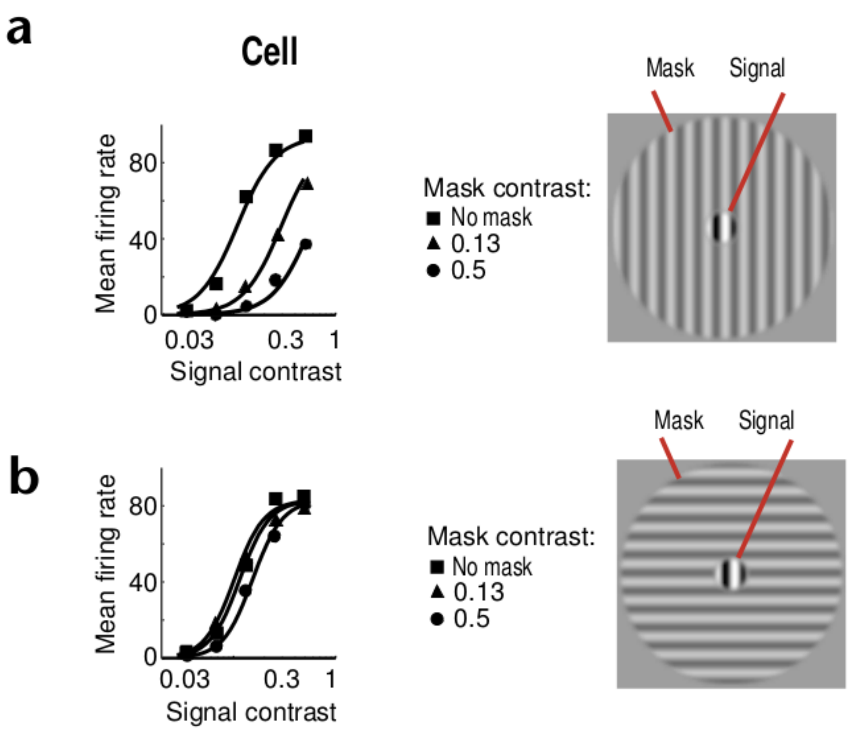

Figure 1. Suppression of responses to optimal stimuli by masking stimuli in primary visual cortex.