VAEs, Population Coding

Question 1: Variational Autoencoders

[6 marks] The KL divergence between the random variables Q and P is

$$\mathrm{KL}\Big(Q \,\big\|\, P\Big) = \int q(x) \ln\left(\frac{q(x)}{p(x)}\right) dx$$

where $q(x)$ and $p(x)$ are the probability density functions for Q and P, respectively.

Put your integration gloves on. Using the definition of KL divergence, prove that

$$\mathrm{KL}\Big(\mathcal{N}(\mu,\sigma) \,\big\|\, \mathcal{N}(0,1)\Big) = -\ln\sigma + \frac{\sigma^2+\mu^2}{2} - \frac{1}{2}$$

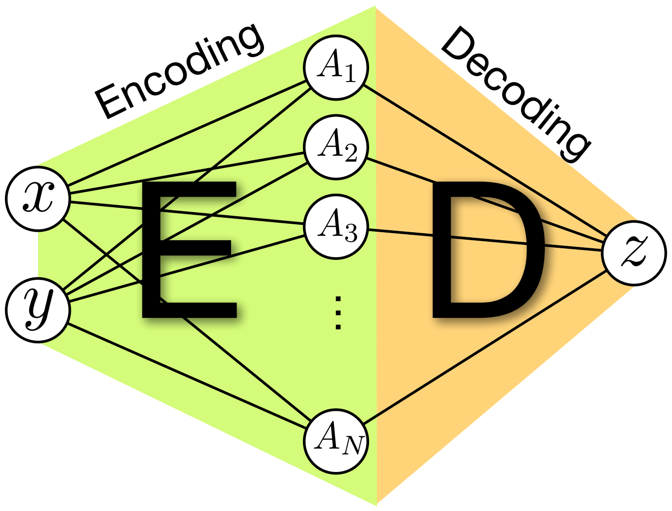
Question 2: Implementing Population Coding

Download the jupyter notebook `a07_YOU.ipynb`. Note that this notebook requires `Network.py`. The notebook defines two classes, `Encoding`, and `Decoding`. The `Encoding` class is complete, but the `Decoding` class needs you to make the finishing touches.

The `Encoding` class is a `Layer` class that takes some input value(s) and represents them in the collective activity of a population of neurons. It consists of a `Connection` object [1], as well as an activation function from the `Network` module (eg. `Logistic`, `Tanh`, `LIF`).

The `Decoding` class is a `Layer` class that takes, as input, the activities of a population of neurons, multiplies them by decoding weights, and outputs the decoded value(s). Like the `Encoding` class, the `Decoding` class contains a `Connection` object to store the decoding weights (biases are all zero). When you create a `Decoding` object, you must specify the `Encoding` population it is decoding from; this is because the decoding weights depend on how data is encoded.

The figure below illustrates how these two `Layer` classes work together to encode and decode data in neural populations.



[1] Recall that a `Connection` object is a type of `Layer` that represents a set of connection weights and biases.

A. [6 marks] Complete the method `compute_decoding_weights` in the `Decoding` class. That method takes a function as input and computes the optimal linear decoding weights that decode the given function values from the neural activities of its `Encoding` population. It stores those decoding weights in the member variable `D`, which is a `Connection` object.

B. [5 marks] Demonstrate that your code works on 1D data by creating a small network that encodes $x \in [-5, 5]$ into the activity of 20 logistic neurons, and then decodes $\operatorname{sinc} x$. You can use NumPy's `sinc` function. Plot x vs $\operatorname{sinc} x$, and x vs z (the output of your network) for at least 200 different x values in the domain.

   For this question, you can either embed the layers in a `Network` object, or feed the data through the layers yourself.

C. [5 marks] Demonstrate that your code works for 2D encodings on a small network. Your network should encode $\theta \in [-\pi, \pi]$ into the activity of a population A containing 80 LIF neurons. If a is the value encoded in A, your network should then decode $(\cos a, \sin a)$, then encode that 2-vector into a population B of 300 LIF neurons. If (b_1, b_2) is the vector encoded in B, your network should, finally, decode $\max(b_1, b_2)$, which should be the output of your network. Each `Encoding` layer should use the supplied `LIF` activation function; you do not need to implement spiking LIF neurons.

   For this question, you must create a `Network` object and add the layers to the network.

   Create a plot of θ vs the output of your network. Also plot θ vs $\max(\cos\theta, \sin\theta)$ on the same axes.

What to submit

Question 1: You can:

- typeset your solutions using a word-processing application, such as Microsoft Word, LaTeX, Google docs, etc., or
- write your solutions using a tablet computer, or
- write your solutions on paper and take photographs.

In any case, it is **your responsibility** to make sure that your solutions are sufficiently legible.

Submit your solution to Crowdmark only. Crowdwark will accept PDFs or image files (JPEG or PNG). You may submit multiple files for each question, if needed.

Question 2: You must submit your jupyter notebook in two places: **Crowdmark**, and **D2L**.

Crowdmark: Export your jupyter notebook as a PDF, and submit the PDF to Crowdmark.

D2L: Submit your `ipynb` file to Desire2Learn in the designated dropbox. You do not need to zip the file.

Make sure you submit your solutions, and not just a copy of the supplied skeleton code. We suggest you rename the `ipynb` file by replacing "`YOU`" with your WatIAM ID (eg. `a07_jorchard.ipynb`).

Important note about submitting code
We re-run your notebook in its entirety. So make sure that all your code cells run, from top to bottom. Any code cell that crashes will stop us from re-running your notebook (and you don't want that). Also, we extract the class and function definitions and autotest them. To facilitate this extraction process, please ensure that:

- all `import` commands are in a single code cell (at the top of the notebook), and
- code cells that contain function definitions and/or class definitions should not include any lines of code that are not part of the definition.