Batch Norm, and Hopfield Networks

Question 1: Continuous Hopfield Networks

[4 marks] A continuous Hopfield network is a continuous-time, recurrent neural network with symmetric connection weights. Let u_i be the input current for neuron $i \in \{1, \ldots, N\}$, and let $v_i = \sigma(u_i)$ be its activity, where $\sigma(\cdot)$ is a non-decreasing activation function ($\sigma'(u) \geq 0,\ \forall u \in \mathbb{R}$).

We define the network's energy as

$$E(u) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij} v_i v_j \ + \ \sum_{i=1}^{N} b_i v_i \tag{1}$$

where $W_{ij} = W_{ji}$ and $W_{ii} = 0$. Each neuron in the network evolves in time according to the differential equation

$$\frac{du_i}{dt} = -\frac{\partial E}{\partial v_i} \qquad \text{for} \quad i = 1, \ldots, N\ . \tag{2}$$

(a) Using equations (1) and (2), derive an expression for $\frac{du_k}{dt}$ in terms of W, v, and b.

(b) Prove that the network energy, E, is non-increasing. In other words, prove that $\frac{dE}{dt} \leq 0$.

Question 2: Implementing Batchnorm

Download the jupyter notebook `a06_YOU.ipynb`. Note that this notebook requires `utils.py`, PyTorch (`torch`), and `tqdm` (find out more here).

In this question, you will implement batch normalization ("batchnorm"), and run learning experiments to see how batchnorm helps a neural network to learn faster.

A. [3 marks] The notebook contains the class `BatchNorm`. **Complete the implementation of the `BatchNorm` class** so that it normalizes the batch. In particular, if $\mathbf{x} \in \mathbb{R}^{D\times N}$ is a batch of inputs, where we denote the dth sample as $\mathbf{x}_d \in \mathbb{R}^N$, then the **output** will be $\mathbf{y} \in \mathbb{R}^{D\times N}$,

$$\mathbf{y}_d = \frac{\mathbf{x}_d - \mu}{\sqrt{\sigma^2 + \varepsilon}}\ , \quad d = 1, \ldots, D\ , \tag{3}$$

where d indexes the samples in the batch, $\mu \in \mathbb{R}^N$ and $\sigma \in \mathbb{R}^N$ are the means and standard deviations (respectively) of the nodes for the batch, and ε is a small, positive constant.

B. [3 marks] **Demonstrate that your `BatchNorm` class works** by running it on the `UClasses` dataset. That is, pass a batch through it and show that each node of the output has zero mean and unit variance, different from the mean and variance of its input. **Include print statements etc. to make your results clear to the reader.**

C. [9 marks] **Add code to the notebook that runs an experiment** to establish the benefit of batchnorm. The experiment should run a series of learning trials on two different, but comparable neural networks. To do so, you should derive 2 different network classes from the supplied `NNBase` class:

- `NormalNet`: A simple neural network that does *not* use batchnorm.
- `BNNet`: A simple neural network that *does* use batchnorm.

Use `LeakyReLU()` as your activation function for the hidden layers (you can use the default slopes), and apply the batchnorm (when appropriate) to the output of the activation function, before the connection to the next layer. Note that you have to define `loss_fcn` in the constructor for each class.

Your goal for this question is to characterize the benefit of batchnorm by summarizing the performance of the two methods. You should run several trials (at least 10) for each method, each time using a new instance of the dataset, and a new instance of the network class. After all your trials, you should **generate a plot** like the one shown in Fig. 1. Each line shows the average loss at each epoch, and the shaded region shows one standard deviation in either direction; the plot uses `matplotlib`'s `fill_between` function. Make sure you plot is labelled. Your plot might look different, depending on the hyperparameters you choose.

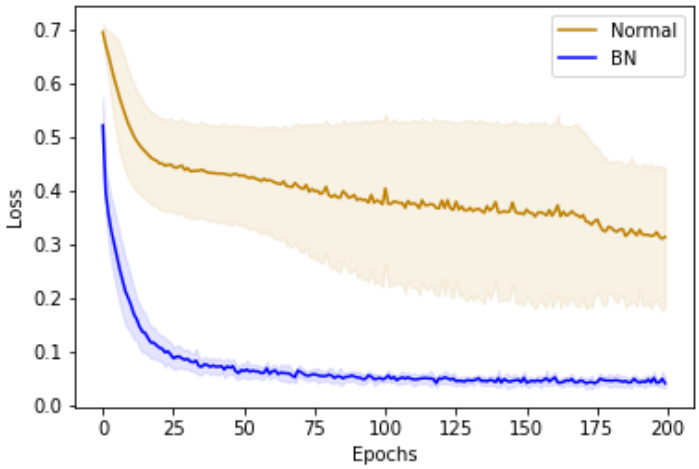

Figure 1: Learning curves for the two different models.

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. `a06_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.