Final Project

CSE 599N: Machine Learning for Neuroscience
Professor Matt Golub
Due: **Wednesday** June 5, 2024 11:59pm

Please review all homework guidance posted on the website before submitting to Gradescope. Reminders:

- All code must be written in Python and all written work must be typeset (e.g. LaTeX).
- Please provide succinct answers and supporting reasoning for each question. Similarly, when discussing experimental results, concisely create tables and/or figures when appropriate to organize the experimental results. All explanations, tables, and figures for any particular part of a question must be grouped together.

Important: By turning in this assignment (and all that follow), you acknowledge that you have read and understood the collaboration policy with humans and AI assistants alike: `https://courses.cs.washington.edu/courses/cse599n/24sp/assignments/`. Any questions about the policy should be raised at least 24 hours before the assignment is due. There are no warnings or second chances. If we suspect you have violated the collaboration policy, we will report it to the college of engineering who will complete an investigation. Not adhering to these reminders may result in point deductions.

Instructions

For your Final project, you will have two options: (1) Guided Project and (2) Open-ended Project. The Guided Project is similar in nature to the coding problems from the previous homeworks. It uses neural data that you have previously seen in HW2 and HW3, and will explore variational autoencoders. The Open-ended Project is for students who want to design their own project. See below for more instructions specific to each option.

Guided Project

Note: The Guided Project is an **individual assignment**; you must independently write up your solutions and adhere to the course policy as you would on any other assignment.

For the Guided Project, we have provided you with the following files:

- `project.pdf` (this file)
- `Guided Project Starter Code` (link)
- `data.zip` (file containing data required for the assignment)

Fill out each of the cells in the notebook that are labeled with "YOUR CODE HERE." You can make the notebook self-contained with all of your code in it, or feel free to write additional Python code which you import and call from your notebook.

Submission: Submit all of your code files to the "Guided Project - Code" assignment on Gradescope. Convert your `.ipynb` file(s) to a single PDF and submit to the "Guided Project - PDF" assignment on Gradescope. Please tag all pages that correspond to each question for the PDF portion of the assignment. Please **do NOT** submit anything to the "Open-ended Project" assignments on Gradescope.

Guided Project Description

Your goal is to implement a variational autoencoder (VAE) and apply it to the real neural data you first encountered in HW2. As a reminder, the neural data were recorded using a 100-electrode array in premotor cortex of a macaque monkey[1] performing the delayed reach task that we have discussed in class. The datasets are in the files "data/ps2a_train.npy" and "data/ps2a_test.npy". The train.npy file contains data you will use to train your models, and the test.npy file contains the data you will use to test your models.

Each variable is a numpy array of dimensions (8 reaching angles) $\times$ (91 trials) $\times$ (97 neurons). Each array is derived from spike trains recorded simultaneously from 97 neurons while the monkey reached 91 times along each of 8 different reaching angles. We have provided you with the spike counts for each neuron within a single 200 ms bin starting 150 ms after the reach target turns on. We ignore those first 150 ms because that is roughly the time it takes time for information about the reach target to arrive in premotor cortex (due to the time required for action potentials to propagate and for visual processing). The number of spikes recorded during the 200 ms window from the ith neuron on the nth trial of the kth reaching angle is `train[k, n, i]`, where $n = 1, \ldots, 91, k = 1, \ldots, 8$, and $i = 1, \ldots, 97$. The same format is used in `test`. We have also reshaped the training and test data for you in the provided starter code to both be of shape $((8 \times 91), 97)$, to simplify using PyTorch to build and train your models. You will find the training data and corresponding labels in the variables `train_data` and `train_labels`, and the test data and labels in

[1] The neural data have been generously provided by the laboratory of Prof. Krishna Shenoy at Stanford University. The data are to be used exclusively for educational purposes in this course.

`test_data` and `test_labels`.

1. Building a VAE
   - (a) [**5 points**] Implement a basic multilayer perceptron that can be used for the encoder and decoder blocks of your VAE. Your code should be flexible, allowing input arguments to specify the number of hidden layers, the size of each hidden layer, and the choice of activation function.
   - (b) [**20 points**] Implement a VAE class. You will need to define the prior distribution of the model, the encoder and decoder blocks of the model, the `encode` and `decode` functions, the `log_like` and `kl_divergence` functions, and the `forward` function.
   - (c) [**5 points**] Implement the main training loop for your VAE. Write a function which iterates through a dataset and trains the model over a specified number of epochs. Allow for gradient-norm clipping, like was done in HW4 (we suggest clipping at a value of 1.0).
   - (d) [**5 points**] Create an instance of a VAE using your class. Then create an optimizer (use Adam) and pass the parameters of the model as an argument. Finally, create a dataset using the provided dataset class and data. Wrap this dataset in a 'DataLoader' (as in HW4), which will handle batching of the data. The specific architecture details of the VAE are up to you. A good starting point would be encoder and decoder sizes (`enc_size` and `dec_size`) of `[32,]`, a latent embedding dimensionality (`hidden_size`) of 4, batch size of 32, and learning rate of 1e-3. You should **not** modify `kl_beta` = 1.0.
   - (e) [**0 points**] Train your model using your training loop.
   - (f) [**5 points**] Plot the loss, log likelihood, KL divergence, and gradient norm throughout training. Create four plots with "Epoch" on the x-axis and "Loss", "LL", "KL", and "Gradient Norm" on the y-axes, respectively. For the loss, use a log scale for the y-axis.
   - (g) [**10 points**] Visualize the encoded datapoints. Use the trained VAE's `encode()` function to get the means of the approximate posteriors. If your VAE has a latent dimensionality (`hidden_size`) greater than 2 (we recommended 4 above), apply PCA to further reduce the dimensionality down to 2. Visualize these 2-dimensional embeddings on a scatter plot. Color each dot appropriately according to reaching angle (there should be a total of 728 dots).
2. [**15 points**] Building a Simple Classifier. Define a SimpleClassifier class (PyTorch `nn.Module`) that is a (relatively) small neural network. One starting point would be two linear layers with a ReLU activation applied between the layers. Then, complete the training loop and evaluation code.
3. Comparing Training Data
   - (a) [**5 points**] Dataset preparation. First, we will need to set up the datasets and DataLoaders for the comparison between the raw data and encoded embeddings. You can reuse the provided `SimpleDataset` class for PyTorch Datasets.
   - (b) [**5 points**] Train a SimpleClassifier model using the raw data. Plot the training loss curve with epochs on the x-axis and loss on the y-axis.
   - (c) [**5 points**] Train a SimpleClassifier model using the encoded data. Plot the training loss curve with epochs on the x-axis and loss on the y-axis.
   - (d) [**5 points**] Evaluate and compare classifiers. How many parameters do each of your trained models include? Which of the models perform better? Why might this be the case?
4. [**20 points**] Building upon a HW2 Classifier. Choose one (or more) of the classifiers you implemented for HW2, Question 4. Try applying this classifier to the encoded data. How does training the classifier on top of the VAE compare to training it on just the raw data? You may change the VAE hyperparameters above. Please list out any important hyperparameters or training details when comparing the classifier trained on top of the VAE to the classifier trained on just the raw data.

Open-ended Project

For students who would like to define and design their own project. The open-ended project will likely be more time and thought intensive compared to the guided project. Some options could involve:

- Exploring models we've discussed in class but on new neuroscience datasets.
- Comparing models we've discussed to other models of your choice, on neuroscience-related datasets.
- Developing new machine learning models with at least some relevance to a neuroscience problem.
- Theory-related explorations with at least some relevance to a neuroscience problem.

If you choose the open-ended project, you may choose to work in a group of up to 3 people, or you may choose to work on your own. Project expectations will scale linearly – we expect a 3-person project to reflect 3x more effort than we would expect from a 1-person project. If you are unsure about any aspects about your project, please reach out to course staff.

Data sources: You may choose to work with datasets from our earlier HW assignments. You may choose to work with publicly available datasets. A few great resources that have each compiled several well-documented datasets include Pei*, Ye* et al., "Neural Latents Benchmark '21, " NeurIPS 2021 and the Neuromatch Academy Computational Neuroscience course. Alternatively, you may use neural data from your graduate research lab or a colleague's lab, or you may generate and use synthetic data if that is most appropriate.

After completing your project, you will write a **Final Project Report** describing the problem you selected and the progress you made. Reports should be single-spaced with 11-point font and 1-inch margins. Reports should not exceed 8 pages including figures and references. Your report should include, at a minimum, the following sections:

- *Introduction*: A brief overview of the scientific and/or methodological problems you addressed in your project.
- *Data Description*: A description of the data used in your project. Please include all of the details needed to evaluate your project (do not rely on references to cover critical details).
- *Methods*: A detailed description of the machine learning models and methods used in your project.
- *Results*: A detailed presentation of your results, which may include model fitting diagnostics, performance metrics, and visualizations. If you compared your results to previous work, include a discussion of the differences and similarities.
- *Discussion*: The implications of your results, including any limitations of your approach and possible avenues for future research.
- *Conclusion*: A summary of your project and its contributions to the fields of neuroscience and machine learning.
- *Author Contributions*: If you worked in a group, please describe each group member's contribution in 1-2 sentences per group member.
- *Acknowledgments*: Please acknowledge any resources that supported your project. This may include code, software, data, advising, or helpful conversations with colleagues.
- *References*: A list of all the sources cited in your report.

Submission: Submit all of your code files to the “Open-ended Project - Code” assignment on Gradescope, and your project report to the “Open-ended Project - Report” assignment on Gradescope. Please tag the pages corresponding to each required section for your report. If you are working in a group, have one person submit to Gradescope and add the other individuals to the assignment. If done correctly, each person in the group should be able to see the submission on their Gradescope page. Please **do NOT** submit anything to the “Guided Project” assignments on Gradescope.