CS689: Machine Learning - Fall 2024

Homework 4

Assigned: Tuesday, November 19th, 2024

Getting Started: This assignment consists of two parts. Part 1 consists of written problems, derivations, and implementation warmup questions, while Part 2 consists of implementation and experimentation problems. You should first complete Part 1. You should then start on the implementation and experimentation problems in Part 2 of the assignment. The implementation and experimentation problems must be coded in Python 3.8+. Download the assignment archive from Canvas and unzip the file. The data files for this assignment are in the `data` directory.

Submission and Due Dates: You may use at most one late day for Part 1 and at most two late days for Part 2. Work submitted after you have used all available late days does not count for credit. You must submit a PDF document to Gradescope with your solutions to Part 1. You must submit a PDF document to Gradescope containing the results of your experiments for Part 2. You must also submit your code and prediction files to Gradescope for Part 2. You are strongly encouraged to typeset your PDF solutions using LaTeX. The source of this assignment is provided to help you get started. You may also submit a PDF containing scans of *clear* hand-written solutions. Work that us illegible will not count for credit.

Academic Honesty Reminder: Homework assignments are individual work. Being in possession of another student's solutions, code, code output, or plots/graphs for any reason is considered cheating. Sharing your solutions, code, code output, or plots/graphs with other students for any reason is considered cheating. Copying solutions from external sources (books, web pages, etc.) is considered cheating. Use of AI tools (e.g., ChatGPT, etc.) in developing answers to both written questions and coding questions is considered cheating. Collaboration indistinguishable from copying is considered cheating. Posting your code to public repositories like GitHub (during or after the course) is not allowed. Manual and algorithmic cheating detection are used in this class. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

Part 1: Derivations and Basic Coding (Due Tuesday, Nov 26th at 11:59pm)

1. (*20 points*) (L4) Suppose we have a data set of $(\mathbf{x}, y)$ pairs where $\mathbf{x} \in \mathbb{R}^D$ and and $y \in \mathbb{R}^{\geq 0}$ (the non-negative reals) and we wish to construct a probabilistic regression model. Answer the following questions.

a. (*5 pts*) What unconditional parametric probability density function $p_\phi(Y = y)$ could you use to model y? Give a mathematical expression for the density function and explain your choice.

b. (*5 pts*) What constraints are there on the parameters of the unconditional parametric probability density function that you selected?

c. (*5 pts*) Using the unconditional parametric probability density function that you selected, construct a probabilistic regression model $p_\theta(Y = y|\mathbf{X} = \mathbf{x})$. Specify mathematical expressions for the conditional density and the parameter prediction functions. Explain your answers.

d. (*5 pts*) Write down and expand the negative log likelihood function for your model.

2. (*10 points*) (L4) Consider the product of normal marginals distribution specified below when answering this question. Assume that $\mathbf{x} = [x_1, ..., x_D] \in \mathbb{R}^D$.

$$p_\theta(\mathbf{X} = \mathbf{x}) = \prod_{d=1}^{D} \mathcal{N}(x_d; \mu_d, \sigma_s^2)$$

a. (*5 pts*) Let $k < D$. Use the definition of marginalization to derive an expression for the marginal distribution on $\mathbf{x}_{1:k} = [x_1, ..., x_k]$. Simplify the answer as much as possible. (Note: your answer must argue algebraically starting from the definition of the joint distribution and should not use probabilistic independence properties.)

b. (*5 pts*) Let $k < D$. Use the definition of marginalization and conditioning to show that the conditional distribution of x_D given $\mathbf{x}_{1:k}$ is equal to the marginal distribution of x_D. (Note: your answer must argue algebraically starting from the definition of the joint distribution and should not use probabilistic independence properties.)

3. (*20 points*) (L4, L8, L9) In this question you will derive and implement computations for inference with mixture models. A full covariance Gaussian mixture model is defined as shown below for $z \in \{1, ..., K\}$ and $\mathbf{x} \in \mathbb{R}^D$. In the equations below, $\mathbf{x}$ and μ_z are assumed to be row vectors. The notation $|A|$ indicates the determinant of the matrix A.

$$\begin{aligned} p_\theta(\mathbf{X} = \mathbf{x}, Z = z) &= p_\theta(Z = z)p_\theta(\mathbf{X} = \mathbf{x}|Z = z) \\ p_\theta(Z = z) &= \pi_z \\ p_\theta(\mathbf{X} = \mathbf{x}|Z = z) &= \frac{1}{|2\pi\Sigma_z|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_z)\Sigma_z^{-1}(\mathbf{x} - \mu_z)^T\right) \end{aligned}$$

A Gaussian mixture model trained on the MNIST data set using the known digit classes as the mixture component indicators is provided in the file `mixture_model.npz`. As a result of this training, there is a one-to-one correspondence between clusters and digit classes. The model consists of a NumPy array of cluster means mu of shape $(10, 784)$, a NumPy array of cluster covariance matrices Sigma of shape $(10, 784, 784)$, and a NumPy array of mixture proportions pi of shape $(10, 1)$. The mean for cluster k is given by mu[k,:], the covariance matrix for cluster k is given by Sigma[k,:,:]. The file `mixture_data.npz` contains 10 example data cases in the array X. The n^{th} data case is given by X[n,:]. Both the mean vectors and data cases can be converted back into single channel images using reshape(28,28). Use this information to answer the following questions.

a. (*5 pts*) Provide a numerically stable, vectorized PyTorch implementation of the inference computation for this mixture model. This computation should take as input a tensor of shape $(N, 784)$ and output a tensor of shape $(N, 10)$ containing the log probabilities $\log(P_\theta(Z = z|\mathbf{X} = \mathbf{x}_n))$ for each n and z. Explain your implementation with supporting equations. (Note: your implementation must be "from scratch" and not use an existing implementation of the Gaussian density or related functions.)

b. (*4 pts*) Provide a table showing the output of the computation $\log(P_\theta(Z = z|\mathbf{X} = \mathbf{x}_n))$ when run on the 10 supplied data cases. The table should have one row per data case, one column per value of z and all log probabilities should be given to four decimal places.

c. (*5 pts*) Suppose we are given a corrupted MNIST image where the pixels on the right half of the image are known to be invalid. Let l represent the indices of the pixels in the left half of the image and $\mathbf{x}^l$ represent the pixel values in the left half of the image. Provide a numerically stable, vectorized PyTorch implementation of the inference computation $\log(P_\theta(Z = z|\mathbf{X} = \mathbf{x}_n^l))$. Explain your implementation with supporting equations. (Note: your implementation must be "from scratch" and not use an existing implementation of the Gaussian density or related functions.)

d. (*4 pts*) Provide a table showing the output of the computation $\log(P_\theta(Z = z|\mathbf{X} = \mathbf{x}_n^l))$ when run using only the left half of the 10 supplied data cases. The table should have one row per data case, one column per value of z and all log probabilities should be given to four decimal places.

e. (*2 pts*) Explain the trends that you see when inference is based on the left half of the images instead of the full images.

Part 2: Model Design and Experimentation (Due Friday, Dec 6th at 11:59pm) In this part of the assignment you will implement and experiment with generative models for music. This problem uses a data set derived from the Lakh MIDI Dataset. MIDI is a standard for representing multi-instrument musical scores symbolically in terms of a sequence of note on/off events with additional attributes. For this problem, we have converted a collection of MIDI files into a simplified PyTorch representation where each song includes the score for a single instrument.

Song m in the data set is represented as a torch tensor $\mathbf{x}_m$ with L_m rows and 4 columns. Each row corresponds to a note on/off event. The first column represents the time since the last event. These values are integers greater than or equal to 0. The second column represents the duration that the note will be played for. These values are also integers greater than or equal to 0. The third column represents the value of the note. There are 128 different possible note values represented as integers from 0 to 127. The fourth column represents the volume level at which the note should be played. These are also values from 0 to 127.

The problem that you will investigate is next note event prediction. Based on a context window of the previous $C = 64$ note events, you will develop a model for predicting a probability distribution over the attributes of the next note event including the time t since the previous note event, the duration d the next note will be played for, the value of the next note n, and the volume v the note should be played at. After training, this model can be autoregressively prompted to generate an output sequence of any length corresponding to a completion of the musical score represented by the prompt.

4. (*5 points*) (L9) To begin, you will to implement a torch data loader for this task (see `torch.utils.data.DataLoader`. You can use any torch methods to construct the data loader. The training data are stored in a directory structure with one folder per artist containing all available training songs by that artist. The training songs are of different lengths. You will need to decide how to extract individual training instances consisting of the note events in the context window and the next note event that is to be predicted. You could extract one instance per song or multiple instances per song. When extracting multiple instances, the context windows could be overlapping or non-overlapping. As your answer to this question, include the code you used to load the training data files and create the torch data loader. Explain your design choices and how you implemented them.

5. (*20 points*) (L4) You will next need to design a model for this problem. Your model must be able to take as input a tensor $\mathbf{x}$ of shape $(C, 4)$ where C is the context window size and 4 is the dimension of each note event as described above. Your model will need to predict a probability distribution over the attributes of the next note including the time since the last note event t, the note duration d, the note value n and the note volume v. We will use a product of marginals consisting of normal distributions over the time since the last note event $p(T = t|\mathbf{x}) = \mathcal{N}(t; \mu^t(\mathbf{x}), (\sigma^t(\mathbf{x}))^2)$, the duration the note should be played $p(D = d|\mathbf{x}) = \mathcal{N}(d; \mu^d(\mathbf{x}), (\sigma^d(\mathbf{x}))^2)$ and the volume the note should be played at $p(V = v|\mathbf{x}) = \mathcal{N}(v; \mu^v(\mathbf{x}), (\sigma^v(\mathbf{x}))^2)$. The probability distribution over the note value will be represented as a discrete distribution $P(N = n|\mathbf{x}) = \pi_n(\mathbf{x})$. Your model must produce as output the parameters of these distributions in a tensor of shape $(1, 134)$ matching this specification:

$$[\mu^t(\mathbf{x}), \sigma^t(\mathbf{x}), \mu^d(\mathbf{x}), \sigma^d(\mathbf{x}), \log(\pi_0(\mathbf{x})), ..., \log(\pi_{127}(\mathbf{x})), \mu^v(\mathbf{x}), \sigma^v(\mathbf{x})]$$

You may use any existing torch functionality to create your model with the exception that pre-trained model components are not allowed. You can use any model structure or composition of model structures to address the problem. Your model will be assessed based on the negative log likelihood of its predictions on held out test data (described below) and the creativity of the model design. Answer the following questions.

a. (*10 pts*)In your report, include an architecture diagram for your final model to provide a visual representation of your model's structure and components, as well as the code implementing your model.

b. (*10 pts*)Referencing your architecture diagram and code, describe the architecture of your final model in terms of its structure, the types of components used, and any hyper-parameters. Explain your design choices and what motivated them. If you tried several different types of model structures, explain how you arrived at your final design.

6. (*10 points*) (L6, L2) Implement a train/validation experiment design to learn and assess your model. Explain how you partitioned the given data. Describe your approach to learning your final model. What loss function did you use? What optimizer did you choose? How did you determine how to configure the optimizer in terms of learning rate, number of epochs, batch size, etc? Did you use regularization of some kind? Did you use any input transformations or data augmentations? How did you select hyper-parameters? Mention any other important details of your approach.

7. (*5 points*) (L7) Provide a plot showing the training and validation set performance of your final model as a function of learning epochs. Also provide a table showing the final training and validation set loss for your model at the end of training. Finally, report your total training time, the average time to make a prediction for a single context window, and the details of the hardware you ran your learning experiment on.

8. (*10 points*) (L4) The test data file `test.pt` contains a tensor `Xte` of shape $(M, C, 4)$ where M is the number of test instances and C is the context window size. For each test instance i represented by `Xte[i,:,:]`, use your final learned model to make a probabilistic prediction for the attributes of the next note. The result will be a tensor of distribution parameters of shape $(1, 134)$ for each test instance. Concatenate the parameter prediction vectors into a singe tensor of shape $(M, 134)$ and save the tensor to a file using `torch.save(Xte,"note_predictions.pt")`. Upload the predictions file to Gradescope along with your model implementation (model.py) and experimentation code (`experiments.py`). Your model will be scored in terms of nll and the your grade for this question will depend on the predictive performance your model achieves relative to reference models implemented by the course staff.