CS689: Machine Learning - Fall 2022

Homework 4

Assigned: Wednesday, November 16th, 2022

Getting Started: This assignment consists of two parts. Part 1 consists of derivations, while Part 2 consists of implementation and experimentation. You should first complete the problems in Part 1. You should then start on the implementation and experimentation problems in Part 2 of the assignment. All coding problems must be coded in Python 3.6+. Download the assignment archive from Moodle and unzip the file. The data files for this assignment are in the `data` directory. Code templates are in the code directory. The only modules you are allowed to use in your implementation are those already imported in the code templates.

Submission: You must submit a PDF document with your Part 1 solutions to Gradescope. You are strongly encouraged to typeset your solutions using LaTeX. The source of this assignment is provided to help you get started; however, **please do not submit the question text along with your solutions**. You may also submit a PDF containing scans of clear hand-written solutions. You must submit a PDF document containing the results of your Part 2 experiments to Gradescope. You must separately submit your code and output files Gradescope. Solutions to Part 1 will be released immediately after Part 1 is due. Late work will only be accepted in accordance with the course's late homework policy.

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. 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 is 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 (Due Wednesday, Nov 30th at 11:59pm)

1. (*10 points*) **Products of Marginals:** Suppose we have continuous data $\mathbf{x} = [x_1, ..., x_D]$ and we choose to model $\mathbf{x}$ via a product of univariate normal marginal distributions as shown below where $\theta = [\mu_1, ..., \mu_D, \sigma_1, ..., \sigma_D]$.

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

Prove that the joint distribution $p(\mathbf{X} = \mathbf{x}|\theta)$ is actually a multivariate normal distribution with a diagonal covariance matrix. Specify what the mean and covariance matrix parameters are in terms of the model parameters $\theta = [\mu_1, ..., \mu_D, \sigma_1, ..., \sigma_D]$.

2. (*10 points*) **Gaussian Independence:** Two continuous random variables A and B are said to be probabilistically independent if $p(A = a, B = b) = p(A = a)p(B = b)$. Suppose we model continuous data $\mathbf{x} = [x_1, ..., x_D]$ with a diagonal Gaussian distribution $p(\mathbf{X} = \mathbf{x}|\mu, \Sigma) = \mathcal{N}(\mathbf{x}; \mu, \Sigma)$. Prove that under this model, any pair of random variables X_i and X_j with $i \neq j$ are probabilistically independent.

3. (*20 points*) **Numerical Stability of Mixtures:** Consider the mixture of Bernoullis model where $P(Z = z) = \pi_z$ and $P(\mathbf{X} = \mathbf{x}|Z = z) = \prod_{d=1}^{D} \phi_{dz}^{x_d}(1 - \phi_{dz})^{(1-x_d)}$. Recall that the joint probability of $\mathbf{x}$ and z is given by $P(Z = z)P(\mathbf{X} = \mathbf{x}|Z = z)$.

a. (*5 pts*) This model can have numerical stability issue when D is large. For example, suppose we have a model component z where $\phi_{dz} = 0.5$ for all d. How large can D be before $P(\mathbf{x}|z)$ underflows to 0 assuming we use 32 bit floating point numbers in the computation?

b. (*5 pts*) Is encountering numerical underflow in the computation of $P(\mathbf{X} = \mathbf{x}|Z = z)$ problematic when trying to compute $P(z|\mathbf{x})$? Explain your answer.

c. (*5 pts*) Suppose that we have a Bernoulli mixture model where computing $P(\mathbf{X} = \mathbf{x}|Z = z)$ underflows. Can we still compute valid values for $\log P(\mathbf{X} = \mathbf{x}|Z = z)$? Explain your answer.

d. (*5 pts*)Suppose that we have a Bernoulli mixture model where computing $P(\mathbf{X} = \mathbf{x}|Z = z)$ underflows. Can we still compute valid values for $\log P(\mathbf{X} = \mathbf{x})$? Explain your answer.

4. (*20 points*) **Mixture Models and Missing Data:** Consider the mixture of normals model where $P(Z = z) = \pi_z$ and $p(\mathbf{X} = \mathbf{x}|Z = z) = \prod_{d=1}^{D} \mathcal{N}(x_d; \mu_{dz}, \sigma_{dz}^2)$. Suppose we have a data vector $\mathbf{x}$ where dimension i is not observed. Denote by $\mathbf{x}_{-i}$ the vector of observed values $\mathbf{x}_{-i} = [x_1, ..., x_{i-1}, x_{i+1}, ..., x_D]$ and let $\mathbf{X}_{-i}$ be the corresponding vector valued random variable.

a. (*5 pts*)Using marginalization, derive an expression for $p(\mathbf{X}_{-i} = \mathbf{x}_{-i}|Z = z)$ starting from $p(\mathbf{X} = \mathbf{x}|Z = z)$.

b. (*5 pts*) Using marginalization, derive an expression for $p(\mathbf{X}_{-i} = \mathbf{x}_{-i})$ starting from $P(\mathbf{X} = \mathbf{x}, Z = z)$.

c. (*5 pts*) Using marginalization and conditioning, derive an expression for $p(Z = z|\mathbf{X}_{-i} = \mathbf{x}_{-i})$.

d. (*5 pts*) Explain how we can use this model to make predictions for the missing data dimension $\mathbf{x}_i$ given the values of the observed dimensions $\mathbf{x}_{-i}$. Provide supporting equations.

Part 2: Implementation (Due Wednesday, Dec 7th at 11:59pm)

5. (*40 points*) **Autoencoders:** In this question you will implement and experiment with using autoencoders to de-noise biosignals. The data for this problem are electrocardiogram signals for individual heart beats. Both clean and noisy versions of all of the training data cases are available in the data set. However, only noisy versions of the test data are available. To begin, you will implement a basic linear autoencoder using PyTorch. Recall that a linear autoencoder is defined using a linear encoding function $f(\mathbf{x})$ that maps from the D-dimensional data space to the K-dimensional code space, and a linear decoding function $g(\mathbf{h})$ that maps from the K-dimensional code space to the D-dimensional data space. Your model implementations for this question should be included in ae.py. Your experiments should be included in ae_experiments.py.

- `__init__`: Takes the data dimension D and the length of the latent code K as input. Initializes the model as needed.
- `reconstruct`: Takes a Torch tensor $\mathbf{X}$ of shape (N, D) as input. Uses the current values of the model parameters to compute the estimated reconstruction tensor $\hat{R}$ such that $\hat{R}[n,:] = g(f(\mathbf{X}[n,:]))$.
- `loss`: Takes Torch tensors $\mathbf{X}$ and $\mathbf{X}'$, each of shape (N, D), as input. Computes and returns the mean squared error $\frac{1}{ND}\sum_{n=1}^{N}\sum_{d=1}^{D}(X[n,d] - X'[n,d])^2$.
- `fit`: Takes Torch Tensors $\mathbf{X}$ and $\mathbf{R}$ as input. Learns the model by minimizing the loss (as defined above) between $g(f(\mathbf{X}))$ and $\mathbf{R}$. Any optimizer can be used.

a. (*5 pts*) Instantiate the model using $D = 100$ and $K = 5$. Set the model parameters to the values specified in `ecg_params.npz`. The V array contains the parameters of the encoder. The W array contains the parameters of the decoder. Using these parameters, compute the reconstruction of the first five data cases in the noisy copy of the test data Xte_noise. For each of the five data cases, make a plot showing the noisy data and the reconstruction as time series. Include the plots in your report.

b. (*5 pts*) Instantiate the model using $D = 100$ and $K = 5$. Set the model parameters to the values specified in `ecg_params.npz`. The V array contains the parameters of the encoder. The W array contains the parameters of the decoder. Using these parameters, compute and report the loss using the noisy training data as the $\mathbf{X}$ argument of the loss function, and the clean training data as the $\mathbf{X}'$ argument of the loss function.

c. (*5 pts*) In this question, you will train the model as a classical autoencoder using clean training data as the $\mathbf{X}$ argument to the fit function and clean training data as the R argument of the fit function. Use $D = 100$ and $K = 10$. You will need to configure the optimizer to ensure convergence. Report the final value of the training loss. Also report the loss between the reconstructed noisy training data and the clean training data. This is the training set denoising loss of the trained model.

d. (*5 pts*) In this question, you will train the model as a denoising autoencoder using noisy training data as the $\mathbf{X}$ argument to the fit function and clean training data as the R argument of the fit function. Use $D = 100$ and $K = 10$. You will need to configure the optimizer to ensure convergence. Report the final value of the training loss, which is the denoising loss as defined in the previous question.

e. (*5 pts*) The primary capacity control hyper-parameter in the linear autoencoder is the dimension of the codes K. Describe an experiment to optimize the value of K to maximize denoising performance.

f. (*5 pts*) Implement the experiment described above and provide at least one plot to summarize the results. What value of K do you find to be optimal? What is the training denoising loss using this value of K?

g. (*5 pts*) In this question you will extend the linear autoencoder to a non-linear autoencoder to attempt to improve denoising performance. As your answer to this question, describe the architecture that you implemented. What layer types did you use? How many layers did you use and what are their sizes? What activation functions did you use? How did you learn the model parameters? How did you select hyper-parameters?

h. (*5 pts*) Apply your model to de-noise the noisy test data. For each of the first five test data cases, make a plot showing the noisy data and the reconstruction as time series. Include the plots in your report. Lastly, save the results as a numpy file using the np.savez function: `np.savez("test_denoised.npz",Rhat)` where Rhat is an NxD numpy array containing the de-noised test data cases. Include this file with your Gradescope code submission. During the assignment development period, the autograder will only check to see that the data file has been uploaded, can be opened, and contains an array of the correct shape. Your submission must pass these tests to qualify for points for this question. During the grading period, your results will be scored based on denoising loss using the clean test data. Points will be awarded for this question based on test set denoising performance.