CS689: Machine Learning - Fall 2019

Homework 4

Assigned: Wednesday, Nov 6. Due: Wednesday, Nov 20 at 11:59pm

Getting Started: You should complete the assignment using your own installation of 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 allowable during autograding are those already imported in the code templates.

Deliverables: This assignment has two types of deliverables: a report and code files.

- **Report:** The solution report will give your answers to the homework questions (listed below). The maximum length of the report is 5 pages in 11 point font, including all figures and tables. You can use any software to create your report, but your report must be submitted in PDF format. You will upload the PDF of your report to Gradescope under `HW03-Report` for grading. Access to Gradescope will be enabled one week before the assignment is due. For full credit, all figures must have proper axes, labels, legends, axis ticks, and titles. Any tables must have proper row and/or column headings and titles or captions. You do **not** need to include the text of questions in your report.
- **Code:** The second deliverable is your code. Your code must be Python 3.6 compatible (no iPython notebooks, other formats, or code from other versions of Python). You will upload a zip file (not rar, bz2 or other compressed format) containing all of your code to Gradescope under `HW03-Programming` for autograding. Access to the autograder will be enabled one week before the assignment is due. When unzipped, your zip file should produce a directory called `code`. If your zip file has the wrong structure, the autograder may fail to run. To receive credit for implementation questions, your code must run in Gradescope.

Academic Honesty Reminder: Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. Homework assignments are individual work. Sharing your solutions with other students is considered cheating. Posting your code to public repositories like GitHub is also considered cheating. Collaboration indistinguishable from copying is considered cheating. 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.

Questions:

1. (*20 points*) **Mixture Model EM Derivations:** Classically, mixture models are fit using the EM algorithm. The EM algorithm can be derived from first principles by maximizing the so called Q-function, which provides a lower bound on the log marginal likelihood. This function is shown below. Note that the ϕ_{zn} variables are subject to the constraints $\sum_{z=1}^{K} \phi_{zn} = 1$ for all n, and $\phi_{zn} \geq 0$ for all z and n. The primary model parameters are π and θ. π are the mixture proportions and are subject to the constraints $\sum_{z=1}^{K} \pi_z = 1$ and $\pi_z \geq 0$ for all z. The constraints on the θ variables (if any) depend on the particular form of the mixture components. Use this information to answer the following questions.

$$Q(\mathcal{D}, \theta, \pi, \phi) = \sum_{n=1}^{N} \sum_{z=1}^{K} \phi_{zn} \big(\log P(\mathbf{X} = \mathbf{x} | Z = z, \theta_z) + \log P(Z = z | \pi) - \log \phi_{zn} \big) \tag{1}$$

a. (*10 pts*) Use the method of Lagrange multipliers to maximize $Q(\mathcal{D}, \theta, \pi, \phi)$ with respect to ϕ_{zn} given values for θ and π. Show your work.

b. (*10 pts*)Use the method of Lagrange multipliers to maximize $Q(\mathcal{D}, \theta, \pi, \phi)$ with respect to π given values for θ and ϕ. Show your work.

2. (*40 points*) **Laplacian Mixture Models:** Mixture models can be defined using any density for the mixture components. In this question, we will experiment with a Laplacian mixture model where the mixture component density for each data dimension is given by a Laplace distribution. The model is defined as shown below. Use this model to answer the following questions.

$$P(Z = z | \pi) = \prod_{k=1}^{K} \pi_k^{[z=k]} \tag{2}$$

$$P(\mathbf{X} = \mathbf{x} | Z = z, \theta_z) = \prod_{d=1}^{D} \frac{1}{2b_{dz}} \exp\left(-\frac{|x_d - \mu_{dz}|}{b_{dz}} \right) \tag{3}$$

a. (*5 pts*) Derive the log marginal likelihood for the Laplacian mixture model assuming a data set $\mathcal{D} = \{\mathbf{x}_n | 1 \leq n \leq N\}$ where $\mathbf{x}_n \in \mathbb{R}^D$. Show your work.

b. (*5 pts*) Derive the expression for the posterior distribution over mixture proportions after conditioning on data $P(Z = z | \mathbf{X} = \mathbf{x})$ for this model. Show your work.

c. (*5 pts*) The function $P(Z = z | \mathbf{X} = \mathbf{x})$ has the potential for numerical overflow when implemented naively. As your answer to this question, explain why this is the case and explain how these computation can be performed safely.

d. (*5 pts*) In the next question, you will implement a direct marginal likelihood maximization method for the Laplacian mixture model instead of the EM algorithm. However, the model has constraints on the parameters that need to be accounted for. We will use the unconstrained optimization variables b'_{dz} and π'_z and map them to the original parameters using $b_{zn} = \exp(b'_{zn})$ and $\pi_z = \frac{\exp(\pi'_z)}{\sum_{z=1}^{K} \exp(\pi'_z)}$. Re-write the log marginal likelihood in terms of the unconstrained variables b'_{zn} and π' and their mappings f and g. Provide the updated form as the answer to this question.

e. (*15 pts*) Starting from the provided template (`mixture.py`), implement the Laplacian mixture model. In the `set_params` method, you must take in values of μ, b and π, and use the inverse mappings for f and g to set the values of the unconstrained parameters b' and π'. In the `get_params` method, you must transform the unconstrained parameters b' and π' back to the standard parameters b and π. Use the results you derived above to implement the `predict_proba` function in a numerically stable way. For the fit function, you will directly maximize the unconstrained form of the marginal likelihood that you specified above in terms of the

parameters μ_{dz}, b'_{dz} and π'. You can use NumPy or PyTorch as the basis for this implementation including PyTorch's automatic differentiation methods. While the API for this problem is written in terms of NumPy data structures, you can implement any auxiliary functions and/or classes you need to solve the problem using your chosen approach. As your answer to this question, describe how you implemented the fit method including what library you used, what optimizer you used, how you initialized the model parameters, what optimizer settings you used and how you chose them, and what convergence assessment method or approach to setting the number of iterations you used.

f. (*5 pts*) Using your implementation, learn the model on training data set TR1. Use values of K between 1 and 20. For each value of K, you should learn the model multiple times and take the model with the best log marginal likelihood on the training set to help with the problem of multiple local optima. Provide a single plot showing the log marginal likelihood of the training data as a function of K and the log marginal likelihood of the test data set TE1 as a function of K.

3. (*40 points*) **Mixture Models and Missing Data:** The marginal likelihood principle is a completely general approach to learning models in the presence of incomplete data. This includes randomly missing data as well as latent variables. In this question, we will extend the implementation from the previous question to deal with the case of missing features values during learning and prediction with the Laplacian mixture model. Let $\mathbf{o}$ represent the dimensions that are observed for a given data case and $\mathbf{m}$ represent the dimensions that are not observed. The model can be updated to account for incomplete data by marginalizing it away as shown below. Use this information to answer the following questions.

$$P(Z = z|\pi) = \prod_{k=1}^{K} \pi_k^{[z=k]} \tag{4}$$

$$P(\mathbf{X_o} = \mathbf{x_o}|Z = z, \theta_z) = \int \left(\prod_{d=1}^{D} \frac{1}{2b_{dz}} \exp\left(-\frac{|x_d - \mu_{dz}|}{b_{dz}} \right) \right) d\mathbf{x_m} \tag{5}$$

a. (*5 pts*) The function $P(\mathbf{X_o} = \mathbf{x_o}|Z = z, \theta_z)$ has a simple closed form expression that can be computed in time proportional to the number of observed data dimensions. Derive this expression.

b. (*5 pts*) Combing the above result with Bayes rule, you can now derive an expression for the posterior distribution over mixture components based on incompletely observed data vectors: $P(Z = z|\mathbf{X_o} = \mathbf{x_o})$. Derive this expression.

c. (*5 pts*) An important use of mixture models is as flexible prediction models that can impute missing data values. Let $\mathbf{o}$ represent a collection of data dimensions with observed values and d be a missing dimension. Show how to use the expression $P(Z = z|\mathbf{X_o} = \mathbf{x_o})$ to compute the conditional mean of $P(X_d = x_d|\mathbf{X_o} = \mathbf{x_o})$.

d. (*5 pts*) Another important property of mixture models is that they can also be learned directly from incomplete data. Use your expression for $P(\mathbf{X_o} = \mathbf{x_o}|Z = z, \theta_z)$ to define the marginal likelihood in the case where the data vectors are incompletely observed.

e. (*10 pts*) Generalize your previous implementation of the `fit` method to accommodate learning from

incomplete data using the re-derived marginal likelihood from the previous question. As the API describes, fit can take as input a data matrix $\mathbf{X}$ where missing values are represented by `np.nan` values. Similarly, use the results you derived above to generalize the `predict_proba` method to correctly compute $P(Z = z|\mathbf{X_o} = \mathbf{x_o})$ when data are missing, and implement the `impute` method by predicting the conditional mean of the missing dimensions given the observed dimensions.

f. (*5 pts*) Using your implementation, learn the model on training data set TR2, which includes missing data. Use values of K between 1 and 20. For each value of K, you should learn the model multiple times and take the model with the best log marginal likelihood on the training set to help with the problem of multiple local optima. Provide a single plot showing the log marginal likelihood of the training data as a function of K and the log marginal likelihood of the test data set TE1 as a function of K. How does learning with missing data compare to learning with complete data?

g. (*5 pts*) Lastly, repeat the previous experiment, but instead of evaluating the marginal likelihood of the test data in TE1, use the learned model to impute the missing values in TR2. The test set TE2 contains the missing values in TR2. Use it to compute a mean absolute imputation error for each value of K. Provide a plot showing imputation error versus K. Does the value of K that provides the highest test log marginal likelihood match the value of K that provides the minimum imputation error?