CS689: Machine Learning - Fall 2017

Homework 3

Assigned: Wednesday, Oct 25. Due: Wednesday, Nov 8 at 11:55pm

Getting Started: You should complete the assignment using your own installation of Python 2.7. 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.

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.
- **Code:** The second deliverable is your code. Your code must be Python 2.7 (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-Q1` and `HW03-Programming-Q2` 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. Your code must run within the autograder and complete within the allotted time limits to count for points.

Academic Honesty Statement: Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. 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. (*70 points*) **Mixture Models for Mixed Data:** A common problem in real-world data analysis is learning models from mixed data. In this problem, you will implement a general probabilistic mixture model for learning from mixed data including real values, binary values, categorical values, and counts. We let Z be the mixture indicator random variable. We allow K mixture components. We define four blocks of variables $\mathbf{X} = [\mathbf{X}^R, \mathbf{X}^B, \mathbf{X}^{Ca}, \mathbf{X}^{Co}]$. The corresponding data values are denoted by $\mathbf{x} = [\mathbf{x}^R, \mathbf{x}^B, \mathbf{x}^{Ca}, \mathbf{x}^{Co}]$. We will assume the dimensionality of variables in each block is D^R, D^B, D^{Ca}, D^{Co}. The number of categories for categorical variable d is C_d. The joint distribution of the mixture indicator variable and the data variables is given by:

$$P(\mathbf{X} = \mathbf{x}, Z = z|\theta) = P(Z = z|\theta^M) \prod_{T \in \{R,B,Ca,Co\}} P(\mathbf{X}^T = \mathbf{x}^T|Z = z, \theta_z^T) \tag{1}$$

$$P(Z = z|\theta^M) = \theta_z^M \tag{2}$$

$$P(\mathbf{X}^R = \mathbf{x}^R|Z = z, \theta^R) = \mathcal{N}(\mathbf{x}^R; \mu_z, \Sigma_z) \tag{3}$$

$$P(\mathbf{X}^B = \mathbf{x}^B|Z = z, \theta^B) = \prod_{d=1}^{D^B} (\theta_{dz}^B)^{[x_d^B]} (1 - \theta_{dz}^B)^{[1-x_d^B]} \tag{4}$$

$$P(\mathbf{X}^{Ca} = \mathbf{x}^{Ca}|Z = z, \theta^{Ca}) = \prod_{d=1}^{D^{Ca}} \prod_{c=1}^{C_d} (\theta_{cdz}^{Ca})^{[x_d^{Ca}=c]} \tag{5}$$

$$P(\mathbf{X}^{Co} = \mathbf{x}^{Co}|Z = z, \theta^{Co}) = \prod_{d=1}^{D^{Co}} \text{Poisson}(x_d^{Co}; \theta_{dz}^{Co}) \tag{6}$$

We specify the following prior distribution $P(\theta)$ (up to normalization constants) on the model parameters, which we will use to form regularized/penalized/MAP estimates of the parameters during learning.

$$P(\mu_z) = \mathcal{N}(\mu_z, 0, 100^2 I) \tag{7}$$

$$P(\Sigma_z) \propto \frac{1}{|\Sigma|^{1/2}} \exp(-\frac{1}{2}\text{trace}(0.01 \cdot I \cdot \Sigma_z^{-1})) \tag{8}$$

$$P(\theta_{dz}^B) \propto (\theta_{dz}^B)^{(1/100)} \cdot (1 - \theta_{dz}^B)^{(1/100)} \tag{9}$$

$$P(\theta_{cz}^{Ca}) \propto \prod_{c=1}^{C_d} (\theta_{cdz}^{Ca})^{(1/100)} \tag{10}$$

$$P(\theta_{dz}^{Co}) \propto (\theta_{dz}^{Co})^{(1/100)} \exp(-\theta_{dz}^{Co}) \tag{11}$$

a. (*5 pts*) Derive an expression for the marginal probability $P(\mathbf{X} = \mathbf{x}|\theta)$.

b. (*5 pts*) Derive an expression for the posterior distribution $P(Z = z|\mathbf{X} = \mathbf{x}, \theta)$.

c. (*5 pts*) Explain how the log marginal likelihood $\sum_{n=1}^{N} \log P(\mathbf{X} = \mathbf{x}_n|\theta)$ and log posterior $\log P(Z = z|\mathbf{X} = \mathbf{x}, \theta)$ can be computed in numerically stable ways using the log-sum-exp trick.

d. (*10 pts*)Starting from the regularized lower bound on the log marginal likelihood given by the expression below, derive the EM algorithm for this model.

$$Q(\theta) = \sum_{n=1}^{N} \left(\mathbb{E}_{q_n(z)}[\log P(\mathbf{X} = \mathbf{x}_n, Z = z|\theta)] + \mathbb{H}(q_n(z)) \right) + \log P(\theta) \tag{12}$$

$$q_n(z) = P(Z = z|\mathbf{X} = \mathbf{x}_n, \theta) \tag{13}$$

e. (*10 pts*) Let X_i be a variable in $\mathbf{X}$ and $\mathbf{X}_{-i}$ be the vector of all variables except for X_i. Derive an expression for the posterior predictive distribution $P(\mathbf{X}_i|\mathbf{X}_{-i} = \mathbf{x}_{-i}, \theta)$. (Hint: you will need a different

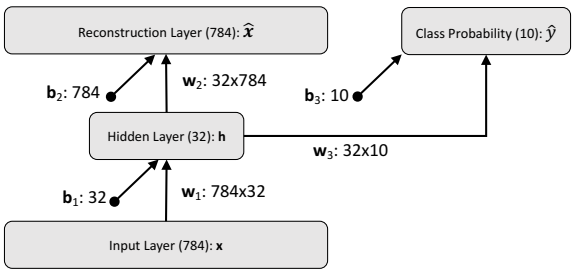
expression for each type of variable).

f. (*20 pts*) Starting from the provided template (mixture.py), implement a class for this model including the functions `fit`, `posterior`, `log_posterior`, `log_maginal_likelihood`, `predict`, `set_model_params`, and `get_model_params`. Note that the predict function will only be tested for cases where the missing dimension is real-valued. For full credit, all of your implementations must use numerically stable computations.

g. (*5 pts*) Use your implementation of the EM algorithm to learn optimal model parameters for $K = 3$ using the Q1 training data. Use 50 EM iterations. Provide a plot of the of the log marginal likelihood as a function of the EM iteration.

h. (*10 pts*) For each test case in Q1 test data, exactly one real variable is missing (specified by a `nan` value). Devise a method for choosing the number of mixture components, learn a mixture model, and use it to make predictions for the missing data values. The value you should predict is the posterior mean of the missing observation: $\mathbb{E}_{P(\mathbf{X}_i|\mathbf{x}_{-i},\theta)}[x_i]$. Use the provided code to save your predictions to the file `q1_prediction.npy` and upload this file to Gradescope for scoring. As your answer to this question, report your final prediction error and explain how you chose the number of mixture components. Note that MSE will be used as the prediction error metric, and that you will likely need to increase the number of learning iterations when increasing K.

2. (*30 points*) **Semi-Supervised Learning with Neural Networks** In this problem, you will experiment with semi-superivsed learning using neural networks. The network architecture is specified below. The hidden layer $\mathbf{h}$ will use 32 units with ReLU non-linearity and is fully connected to the inputs $\mathbf{x} \in \mathbb{R}^D$. There are two output layers. The first output layer, $\hat{\mathbf{x}}$, has the same size as the input $\mathbf{x}$ and will use linear units. The $\mathbf{x} \rightarrow \mathbf{h} \rightarrow \hat{\mathbf{x}}$ path in the model is a basic auto-encoder network. We denote the function corresponding to this half of the network by $\hat{\mathbf{x}} = f^a(\mathbf{x}, \theta)$. The second parallel output layer is a multi-class classification layer that is fully connected to the hidden units and will use a softmax non-linearity to produce a probabilistic class output $\hat{\mathbf{y}}$. The $\mathbf{x} \rightarrow \mathbf{h} \rightarrow \hat{\mathbf{y}}$ path in the model is thus a standard feed-forward neural network classifier. We denote the function corresponding to this half of the network by $\hat{\mathbf{y}} = f^c(\mathbf{x}, \theta)$. We represent the class labels using a one-hot encoding such that $y_c = 1$ indicates that the example belongs to class c.



In semi-supervised learning, the main idea is that the available data include a mix of both labeled and un-labeled examples. To learn the model, we define a composite loss that includes both an auto-encoder component and a classification loss component. Each data case consists of a feature vector $\mathbf{x}_n$ and a one-hot encoded class label y_n. When a class label is not available for a given data case, $y_{cn} = 0$ for all c. The learning problem is given below where α is a parameter that trades off between the auto-encoder loss (mean

squared error) and the classification loss (cross-entropy). In the questions below, you can implement this network using your choice of pytorch (version 0.2.0) or tensorflow (version 1.3).

$$\theta_* = \arg\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} \left(-\alpha \sum_{C=0}^{9} \mathbf{y}_{cn} \log \hat{\mathbf{y}}_{cn} + (1-\alpha) \frac{1}{D} \sum_{d=1}^{D} (\mathbf{x}_{dn} - \hat{\mathbf{x}}_{dn})^2 \right)$$

a. (*5 pts*) Explain in theory how to compute this objective function in a numerically stable way giving equations to support your approach. Next, explain how your approach can be implemented in your chosen framework.

b. (*10 pts*)Starting from the provided template (nn.py), implement a Scikit-Learn compatible class for the model shown above including `objective`, `fit`, `predict_x`, `predict_y`, `set_model_params`, and `get_model_params` functions. As your answer to this question, describe your approach to learning in detail (parameter initialization, optimization algorithm, stepsize selection and convergence rules used, acceleration techniques, etc.), and submit your commented code for auto grading as described above.

c. (*5 pts*) Using the provided Q2 training data set, learn the model using a range of values of α between 0 and 1. Provides plots of the classification error rate and the auto-encoder loss on the training data as a function of α.

d. (*10 pts*)Select a cross-validation approach and use it with the provided training data to optimize as many of the network and learning hyper-parameters as you can (this can include the number of hidden units as well as the number of hidden layers). Use the provided Q2 test data to make classification predictions for all of the training data cases. Use the provided code to save your predictions to the file `q2_prediction.npy` and upload this file to Gradescope for scoring. As your answer to this question, report your best prediction error, describe which hyper-parameters you tried to optimize, and what optimal values you found for them. Support your explanations with appropriately chosen plots.