Topics in Probabilistic Modeling and Inference
(CS698X, Spring 2019)
Homework 2
Due Date: March 14, 2019 (11:59pm)

Instructions:

- Only electronic submissions will be accepted. Your main PDF writeup must be typeset in LaTeX (please also refer to the "Additional Instructions" below).
- Your submission will have two parts: The main PDF writeup (to be submitted via Gradescope `https://www.gradescope.com/`) and the code for the programming part (to be submitted via this Dropbox link: `https://tinyurl.com/cs698x-sp19-hw2`). Both parts must be submitted by the deadline to receive full credit (**delay in submitting either part would incur late penalty for both parts**). We will be accepting late submissions upto 72 hours after the deadline (with every 24 hours delay incurring a 10% late penalty). We won't be able to accept submissions after that.

Additional Instructions

- We have provided a LaTeX template file `hw2sol.tex` to help typeset your PDF writeup. There is also a style file `pmi.sty` that contain shortcuts to many of the useful LaTeX commends for doing things such as boldfaced/calligraphic fonts for letters, various mathematical/greek symbols, etc., and others. Use of these shortcuts is recommended (but not necessary).
- Your answer to every question should begin on a new page. The provided template is designed to do this automatically. However, if it fails to do so, use the `\clearpage` option in LaTeX before starting the answer to a new question, to *enforce* this.
- While submitting your assignment on the Gradescope website, you will have to specify on which page(s) is question 1 answered, on which page(s) is question 2 answered etc. To do this properly, first ensure that the answer to each question starts on a different page.
- Be careful to flush all your floats (figures, tables) corresponding to question n before starting the answer to question $n + 1$ otherwise, while grading, we might miss your important parts of your answers.
- Your solutions must appear in proper order in the PDF file i.e. solution to question n must be complete in the PDF file (including all plots, tables, proofs etc) before you present a solution to question $n + 1$.
- For the programming part, all the code and README should be zipped together and submitted as a single file named `yourrollnumber.zip`. Please DO NOT submit the data provided.

Problem 1 (20 marks)

(Speeding Up Gaussian Processes) Consider Gaussian Process (GP) regression where $y_n = f(\boldsymbol{x}_n) + \epsilon_n$ with f modeled by $\mathcal{GP}(0, \kappa)$ where GP mean function is 0 and kernel/covariance function is κ, and noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$. For simplicity, we will assume the noiseless setting, so $y_n = f(\boldsymbol{x}_n) = f_n$. Given N training inputs $(\mathbf{X}, \boldsymbol{f}) = \{\boldsymbol{x}_n, f_n\}_{n=1}^N$, we have seen that the posterior predictive distribution for a new input $\boldsymbol{x}_*$ is

$$p(f_*|\boldsymbol{x}_*, \mathbf{X}, \boldsymbol{f}) = \mathcal{N}(f_*|\boldsymbol{k}_*^\top \mathbf{K}^{-1}\boldsymbol{f}, \kappa(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^\top \mathbf{K}^{-1}\boldsymbol{k}_*)$$

In the above, $\mathbf{K}$ is the $N \times N$ kernel matrix of training inputs and $\boldsymbol{k}_*$ is $N \times 1$ vector of kernel based similarities of $\boldsymbol{x}_*$ with each of the training inputs. As we know, the above has $O(N^3)$ cost due to $N \times N$ matrix inversion.

Let's consider a way to reduce this cost to make GPs more scalable. To do this, suppose there are another set of *pseudo* training inputs $\mathbf{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M\}$ with $M \ll N$, along with their respective noiseless *pseudo* outputs $\boldsymbol{t} = \{t_1, \ldots, t_M\}$ modeled by the same GP, i.e., $t_m = f(\boldsymbol{z}_m)$. Note again that $(\mathbf{Z}, \boldsymbol{t})$ are NOT known.

Now assume the likelihood for each training output f_n to be modeled by a posterior predictive having the same form as the GP regression's posterior predictive (given above) but with $(\mathbf{Z}, \boldsymbol{t})$ acting as "pseudo" training data.

$$p(f_n|\boldsymbol{x}_n, \mathbf{Z}, \boldsymbol{t}) = \mathcal{N}(f_n|\tilde{\boldsymbol{k}}_*^\top \tilde{\mathbf{K}}^{-1}\boldsymbol{t}, \kappa(\boldsymbol{x}_n, \boldsymbol{x}_n) - \tilde{\boldsymbol{k}}_n^\top \tilde{\mathbf{K}}^{-1}\tilde{\boldsymbol{k}}_n)$$

In the above, $\tilde{\mathbf{K}}$ is the $M \times M$ kernel matrix of the pseudo inputs $\mathbf{Z}$ and $\tilde{\boldsymbol{k}}_n$ is the $M \times 1$ vector of kernel based similarities of $\boldsymbol{x}_n$ with each of the pseudo inputs $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$.

For this problem setup, your goals are the following

1. Derive and write down the expression of the posterior predictive distribution for the output y_* of a new input $\boldsymbol{x}_*$, i.e., $p(y_*|\boldsymbol{x}_*, \mathbf{X}, \boldsymbol{f}, \mathbf{Z})$. Note that here we are assuming that we have estimated the unknown pseudo inputs $\mathbf{Z}$ which this posterior predictive will be conditioned on, in addition to the actual training inputs and outputs $(\mathbf{X}, \boldsymbol{f})$. Note however that the pseudo outputs $\boldsymbol{t}$ will still need to be marginalized out to obtain this posterior predictive, so your derivation must also do that. How does this posterior predictive for y_* compare with the usual GP's posterior predictive for y_* in terms of computational cost?

2. Part (1) requires the pseudo inputs $\mathbf{Z}$. Since we don't know them, we must estimate them from the actual training data $(\mathbf{X}, \boldsymbol{f})$. Let's use MLE-II to estimate $\mathbf{Z}$. In particular, derive the expression for the marginal likelihood $p(\boldsymbol{f}|\mathbf{X}, \mathbf{Z})$. You do not have to show how to solve the optimization problem for MLE-II. Just write down the MLE-II objective.

Note: Feel free to use properties of Gaussians (e.g., marginals from conditionals) to avoid deriving everything from scratch.

Problem 2 (35 marks)

(Two Flavors of EM for an LVM) Suppose we are given N observations $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and we wish to model them via a latent variable model with M mixture component. Assume the following generative story

- For $n = 1, \ldots, N$
  - Draw a mixture component id $c_n \sim$ multinoulli$(\pi_1, \ldots, \pi_M)$. Suppose $c_n = m \in \{1, \ldots, M\}$
  - Generate a K dimensional latent variable $\boldsymbol{z}_n$ from $p(\boldsymbol{z}_n|c_n = m) = \mathcal{N}(0, \mathbf{I}_K)$
  - Generate $\boldsymbol{x}_n$ as $\boldsymbol{x} = \mu_m + \mathbf{W}_m\boldsymbol{z} + \epsilon_n$ where $\epsilon_n \in \mathcal{N}(0, \sigma_m^2\mathbf{I}_D)$.

Essentially, each observation $\boldsymbol{x}_n$ is generated by first drawing $c_n \in \{1, \ldots, M\}$, then generating a latent factor $\boldsymbol{z}_n \in \mathbb{R}^K$, and finally generating the observation $\boldsymbol{x}_n \in \mathbb{R}^D$ as described above. Here $\mu_m \in \mathbb{R}^D$, $\mathbf{W}_m \in \mathbb{R}^{D \times K}$.

The model has the following unknowns $\{c_n, \boldsymbol{z}_n\}_{n=1}^N$ and $\Theta = \{\pi_m, \mu_m, \mathbf{W}_m, \sigma_m^2\}_{m=1}^M$. Treat $\{c_n, \boldsymbol{z}_n\}_{n=1}^N$ as (local) latent variables and $\Theta = \{\pi_m, \mu_m, \mathbf{W}_m, \sigma_m^2\}_{m=1}^M$ as (global) parameters.

Your goal is to derive do point estimation (MLE) for Θ using expectation-maximization (EM). You will consider two flavors of the EM algorithm for this model:

- EM-1: It shouldn't require estimating $\{\boldsymbol{z}_n\}_{n=1}^N$ but only $\{c_n\}_{n=1}^N$ and parameters Θ.
- EM-2: It should estimate $\{c_n, \boldsymbol{z}_n\}_{n=1}^N$ and Θ.

For each of the above two flavors of EM, clearly write down the (1) Conditional posteriors of the latent variables, (2) The complete data log-likelihood (CLL) and the expected CLL, (3) Identify the various expectations needed for the expected CLL and their equations, (4) The M step update equations for Θ, (5) The overall sketch of the EM algorithm, (6) The overall sketch of the corresponding stepwise (online) EM algorithm.

Note: You should skip very detailed derivations for the M step updates of Θ (better to do it using pen and paper and only write the main steps/results in the PDF). Also, if some of the M step updates are obvious or akin to what you may have seen for other simpler models elsewhere, feel free to directly write the final expressions for those.

Problem 3 (15 marks)

(Mean-field VI for Sparse Bayesian Linear Regression) Assume N observations $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ generated from a linear regression model $y_n \sim \mathcal{N}(y_n|\boldsymbol{w}^\top \boldsymbol{x}_n, \beta^{-1})$. Further assume a Gaussian prior on $\boldsymbol{w}$ with different component-wise precisions, i.e., $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|0, \text{diag}(\alpha_1^{-1}, \ldots, \alpha_D^{-1}))$. Also assume gamma priors on the noise precision β and prior's precisions $\{\alpha_d\}_{d=1}^D$, i.e., $\beta \sim \text{Gamma}(\beta|a_0, b_0)$ and $\alpha_d \sim \text{Gamma}(\alpha_d|e_0, f_0), \forall d$. Assume the shape-rate parametrization of the gamma: $\text{Gamma}(\eta|\tau_1, \tau_2) = \frac{\tau_2^{\tau_1}}{\Gamma(\tau_1)} \eta^{\tau_1 - 1} \exp(-\tau_2 \eta)$.

Derive the mean-field VI algorithm for approximating the posterior distribution $p(\boldsymbol{w}, \beta, \alpha_1, \ldots, \alpha_D|\boldsymbol{y}, \mathbf{X})$.

Problem 4 (25 marks)

(VI for Bayesian Logistic Regression) Bayesian logistic regression is a simple yet non-conjugate model (even if we are only interested in getting the posterior of the weights and hyperparameters are held fixed). Given that we have now seen various methods to do VI for non-conjugate models, we will try two of these methods on Bayesian logistic regression, based on the idea of using Monte-Carlo approximation of the ELBO's gradient: (1) Black-box VI based on score-function gradients, and (2) Reparametrization trick based on pathwise gradients.

Assume the training data to consist of N examples $\{\boldsymbol{x}_n, y_n\}_{n=1}^N$ with $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \{-1, +1\}$ modeled as $p(y_n|\boldsymbol{w}, \boldsymbol{x}_n) = \sigma(y_n \boldsymbol{w}^\top \boldsymbol{x}_n)$. Assume the weight vector $\boldsymbol{w}$ to have a prior $p(\boldsymbol{w}) = \mathcal{N}(0, \lambda^{-1}\mathbf{I})$ and the hyperparameter λ to be fixed. Let's assume our variational approximation to be $q(\boldsymbol{w}|\phi) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and assume the positive-definite matrix $\boldsymbol{\Sigma}$ to be modeled as $\mathbf{L}\mathbf{L}^\top$ where $\mathbf{L}$ is a $D \times D$ real-valued matrix.

Derive the expressions for Monte-Carlo gradients of the ELBO w.r.t. $\boldsymbol{\mu}$ and $\mathbf{L}$ using both score-function gradients method (used in BBVI) as well as pathwise gradient method (used in reparametrization trick). You should clearly write down the expressions for all the gradients involved. Also give a brief sketch of the overall VI algorithm for each case. For both cases, you can assume that the Monte-Carlo approximations are computed using S samples and each iteration of the VI algorithm uses a minibatch containing B examples (using $B = 1$ is also fine).

Problem 5 (25 marks)

(Programming Problem) Implement EM for a simplified version of the Gaussian mixture model (GMM) and apply it on a small version of MNIST data (10,000 images). Assume all the Gaussians have a shared spherical diagonal covariance matrix $\sigma^2\mathbf{I}_D$. The MLE solution for σ^2 is $\frac{1}{ND}\sum_{n=1}^{N}\sum_{k=1}^{K}\mathbb{E}[z_{nk}]||\boldsymbol{x}_n - \mu_k||^2$, and the MLE solutions for mixing proportions π_k's and means μ_k's have their usual expressions as in GMM.

We have provided you some skeleton code in MATLAB (if you are using Python, you should be able to easily translate it). The code uses random initialization of the cluster means.

Run your EM algorithm for 200 iteration with $K = 5, 10, 15, 20$ and, for each case, draw the means of each cluster (after reshaping each mean as a 28×28 image), which should somewhat resemble real digits representing the respective clusters.

Now implement online (stepwise) EM for this model with minibatch size = 100, decaying learning rate = $(t+1)^{-\kappa}$ where t is the current iteration number and $\kappa \in (0.5, 1)$ (try 0.55), and repeat the above experiment. For the online EM, the t^{th} iteration's M step updates for each parameter will be of the form $\theta^{new} = (1-\eta_t)\theta^{old} + \eta_t\hat{\theta}$ where $\hat{\theta}$ is the parameter's estimate computed only using the current minibatch of data.

Deliverables: Submit your code for both parts as well as the plots of cluster means.

For practice: Implementing EM for GMM (and also in general) can be tricky if you are doing it for the first time (that's actually the whole point of this exercise :)). I would encourage you to try other variations of the model, e.g., difference covariance matrix for each cluster, non-spherical covariance matrix (diagonal or full), etc. Also, once you have implemented EM, it is actually not that hard to implement other related inference algorithms for GMM, such as VI, SVI, etc., or using EM or such algorithms for other models similar to GMM, such as the model given in Problem 2.