# CS480/680: Introduction to Machine Learning

Homework 4

Due: 11:59 pm, July 23, 2021, submit on CrowdMark.

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TAs can easily run and verify your results. Make sure your code runs!
[Text in square brackets are hints that can be ignored.]

## Exercise 1: Gaussian Mixture Model (GMM) (10 pts)

**Notation**: For a matrix $A$, $|A|$ denotes its determinant. For a diagonal matrix $\text{diag}(\mathbf{s})$, $|\text{diag}(\mathbf{s})| = \prod_i s_i$.

**Algorithm 1:** EM for GMM.

**Input:** $X \in \mathbb{R}^{n\times d}$, $K \in \mathbb{N}$, initialization for *model*
// *model* includes $\pi \in \mathbb{R}_+^K$ and for each $1 \le k \le K$, $\boldsymbol{\mu}_k \in \mathbb{R}^d$ and $S_k \in \mathbb{S}_+^d$
// $\pi_k \ge 0$, $\sum_{k=1}^K \pi_k = 1$, $S_k$ symmetric and positive definite.
// random initialization suffices for full credit.
// alternatively, can initialize $r$ by randomly assigning each data to one of the $K$ components
**Output:** *model*, $\ell$

**for** $iter = 1 : \text{MAXITER}$ **do**
  // step 2, for each $i = 1, \dots, n$
  **for** $k = 1, \dots, K$ **do**
    $r_{ik} \leftarrow \pi_k |S_k|^{-1/2} \exp[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top S_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)]$ // compute responsibility
  // for each $i = 1, \dots, n$
  $r_{i\cdot} \leftarrow \sum_{k=1}^K r_{ik}$
  // for each $k = 1, \dots, K$ and $i = 1, \dots, n$
  $r_{ik} \leftarrow r_{ik}/r_{i\cdot}$ // normalize
  // compute negative log-likelihood
  $\ell(iter) = -\sum_{i=1}^n \log(r_{i\cdot})$
  **if** $iter > 1$ && $|\ell(iter) - \ell(iter-1)| \le \text{TOL} * |\ell(iter)|$ **then**
    **break**
  // step 1, for each $k = 1, \dots, K$
  $r_{\cdot k} \leftarrow \sum_{i=1}^n r_{ik}$
  $\pi_k \leftarrow r_{\cdot k}/n$
  $\boldsymbol{\mu}_k = \sum_{i=1}^n r_{ik}\mathbf{x}_i / r_{\cdot k}$
  $S_k \leftarrow \left(\sum_{i=1}^n r_{ik}\mathbf{x}_i\mathbf{x}_i^\top / r_{\cdot k}\right) - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^\top$

a) (5 pts) Derive and implement the EM algorithm for the diagonal Gaussian mixture model, where all covariance matrices are constrained to be diagonal. Algorithm 1 recaps all the essential steps and serves as a hint rather than a verbatim instruction. In particular, you must change the highlighted steps accordingly (with each $S_k$ being a diagonal matrix), along with formal explanations. Analyze the space and time complexity of your implementation.

[You might want to review the steps we took in class for a simpler case, and ensure you can derive the updates in Algorithm 1. Then adapt the steps to the simpler diagonal case. The solution should look like $s_j = \frac{\sum_{i=1}^n r_{ik}(x_{ij}-\mu_j)^2}{\sum_{i=1}^n r_{ik}} = \frac{\sum_{i=1}^n r_{ik}x_{ij}^2}{\sum_{i=1}^n r_{ik}} - \mu_j^2$ for the $j$-th diagonal. Multiplying an $n \times p$ matrix with a $p \times m$ matrix costs $O(mnp)$. Do not maintain a diagonal matrix explicitly; using a vector for its diagonal suffices. ]

[Warning: Either in this part or the next one, you may run into issues involving NaNs. You will have to diagnose these issues and fix them.]

To stop the algorithm, set a maximum number of iterations (say MAXITER = 500) and also monitor the

change of the negative log-likelihood $\ell$:

$$\ell = -\sum_{i=1}^{n} \log \left[ \sum_{k=1}^{K} \pi_k |2\pi S_k|^{-1/2} \exp[-\tfrac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top S_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)] \right], \tag{1}$$

where $\mathbf{x}_i$ is the $i$-th column of $X^\top$. As a debug tool, note that $\ell$ should decrease from step to step, and we can stop the algorithm if the decrease is smaller than a predefined threshold, say $\text{TOL} = 10^{-5}$.

Run your algorithm on `gmm_dataset.csv`, for $k = 1$ to 10. Generate a plot with $k$ on the x-axis and the negative log-likelihood of the data under the final trained model on the y-axis. What do you think the most appropriate choice of $k$ is? Explain and justify how and why you chose this value. [You may want to focus on more than just maximizing the log-likelihood.] For your chosen value of $k$, report the parameters (mixing weights, mean vectors, and vectors corresponding to the diagonals of the covariance matrices) of your trained model. When reporting them, sort the components in increasing order of mixing weights.

Ans:

b) (5 pts) Next, we apply (the adapted) Algorithm 1 in part a) to the MNIST dataset. For each of the 10 classes (digits), we can use its (only its) training images to estimate its (class-conditional) distribution by fitting a GMM (with say $K = 5$, roughly corresponding to 5 styles of writing this digit). This gives us the density estimate $p(\mathbf{x}|y)$ where $\mathbf{x}$ is an image (of some digit) and $y$ is the class (digit). We can now classify the test set using the Bayes classifier:

$$\hat{y}(\mathbf{x}) = \arg\max_{c=0,\ldots,9} \underbrace{\Pr(Y = c) \cdot p(X = \mathbf{x}|Y = c)}_{\propto \Pr(Y=c|X=\mathbf{x})}, \tag{2}$$

where the probabilities $\Pr(Y = c)$ can be estimated using the training set, e.g., the proportion of the $c$-th class in the training set, and the density $p(X = \mathbf{x}|Y = c)$ is estimated using GMM for each class $c$ separately. Report your error rate on the test set as a function of $K$ (if time is a concern, using $K = 5$ will receive full credit).

[Optional: Reduce dimension by PCA may boost accuracy quite a bit. Your running time should be on the order of minutes (for one $K$), if you do not introduce extra for-loops in Algorithm 1.]

[In case you are wondering, our classification procedure above belongs to the so-called plug-in estimators (plug the estimated densities to the known optimal Bayes classifier). However, note that estimating the density $p(X = \mathbf{x}|Y = c)$ is actually harder than classification. Solving a problem (e.g. classification) through some intermediate harder problem (e.g. density estimation) is almost always a bad idea.]

Ans:

**Exercise 2: VAEs and GANs (10 pts)**

Code provided for this exercise is assuming a PyTorch solution, you may change it as necessary if you prefer to use TensorFlow or JAX. You may find this tutorial for GANs helpful.

a (4.5 pts) Complete the implementation of a VAE in vae.py. In your solution writeup, include the two produced graphs (corresponding to the training and test sets), with the epoch number on x-axis and the loss on the y-axis. Further include the produced samples from every 10th epoch (after epochs 10, 20, 30, 40, and 50).

Ans:

b (4.5 pts) Complete the implementation of a GAN in gan.py. In your solution writeup, include the two produced graphs (corresponding to the training and test sets), with the epoch number on x-axis and the losses (both generator and discriminator) on the y-axis. Further include the produced samples from every 10th epoch (after epochs 10, 20, 30, 40, and 50).

Ans:

c (1 pt) Describe, compare, and constrast your produced results and experiences with GANs and VAEs in this exercise.

Ans: