COMP6321: Machine Learning

Assignment 3

Due: 11:59 PM (EST), October 22, 2024, submit on Moodle.

Include your name and student ID!

- Submit your write-up in PDF (we do not accept hand-written submissions) and all source code as an ipynb file (with proper documentation). Your writeup should include written answers to all questions, including the reported results and plots of coding questions in a single PDF file. Please save the output of each cell or your coding questions may NOT be graded. Write a script for each programming exercise so the TAs can easily run and verify your results. Make sure your code runs!
- Text in square brackets are hints that can be ignored.

NOTE For all the exercises, you are only allowed to use the basic Python, Numpy, and matplotlib (or other libraries for plotting), unless specified otherwise.

Exercise 1: K-means clustering (6 pts)

NOTE: You are allowed to compare your results with the Scikit implementation of k-means, but you cannot use the Scikit implementation as your own solution.
In this exercise, you will be using the MNIST dataset. To work with this dataset, you will first need to flatten your images from 28×28 to 784×1 vectors. Next, use the PCA in scikit learn to convert the 784×1 vectors to 20×1 vectors. You can use the training data for this exercise. Use all the classes in the dataset.

1. (2 pts) Implement the k-means clustering algorithm with euclidean distance as the distance metric.
2. (1.5 pts) Apply your k-means implementation to the MNIST dataset. Use k={5,10,20,40}. Also, you will not be using class labels during the clustering process as it is an unsupervised learning problem.
3. (1 pt) As we do not use class labels for clustering we cannot use test error to evaluate the k-means algorithm. Instead, we can test our clustering implementation using cluster consistency: all data points in a cluster should belong to the same class. For example, if your cluster number 1 has all the data points belonging to class "5" then this cluster has perfect consistency. You can find the cluster consistency Q_i for each cluster i as follows:
$$Q_i = \frac{m_i}{N_i} \tag{1}$$
where, N_i is the total number of data points in cluster i, and m_i represents the total number of data points belonging to the most common class in cluster i. You can find m_i by first counting the total number of data points belonging to each of the 10 classes in cluster i, and then taking the max of this list. To find the overall clustering consistency:
$$Q = \frac{1}{k}\sum_{i=1}^{k} Q_i \tag{2}$$
Report the cluster consistency of your k-means clustering on the MNIST dataset for all four values of k.
4. (1.5 pts) Which k value produces the best results? Explain. Can the results from cluster consistency be misleading? Explain. [HINT Intuitively, what k value should produce the best results on the MNIST dataset?]

NOTE: If timing is an issue, you can use 500 data points for each class.

Exercise 2: Gaussian Mixture Model (GMM) (10 pts)

Notation: For a matrix A, $|A|$ denotes its determinant. For a diagonal matrix $\text{diag}(\mathbf{s})$, $|\text{diag}(\mathbf{s})| = \prod_i s_i$.
Note: For 2.1, you only need to implement (or analyze) your algorithm without testing it on any dataset. Implementations in 2.1 will be tested in 2.2. Therefore, any questions related to the testing of your implementation in 2.1 on a dataset, can be answered in 2.2.

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 $iter = 1 :$ 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 $iter > 1$ && $|\ell(iter) - \ell(iter-1)| \le$ TOL $* |\ell(iter)|$ **then**
    break
  // step 1, for each $k = 1, \dots, K$
  $r_{\cdot k} \leftarrow \sum_{i=1}^n r_{ik}$
  $\pi_k \leftarrow r_{\cdot k}/n$
  $\boldsymbol{\mu}_k = \sum_{i=1}^n r_{ik}\mathbf{x}_i / r_{\cdot k}$
  $S_k \leftarrow \left(\sum_{i=1}^n r_{ik}\mathbf{x}_i\mathbf{x}_i^\top / r_{\cdot k}\right) - \boldsymbol{\mu}_k\boldsymbol{\mu}_k^\top$

1. (6 pts) Derive and implement the EM algorithm for the diagonal Gaussian mixture model (dGMM), where all covariance matrices are constrained to be diagonal. Note that the above algorithm does not make any assumptions about the covariance matrices, however, in our class, we assumed the covariances to be identity matrices. 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..

   [You might want to review the steps we took in class to get the updates in Algorithm 1 and adapt them to a bit more complex case here. 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.]

   To stop the algorithm, set a maximum number of iterations (say MAXITER $= 500$) and also monitor the change of the negative log-likelihood ℓ:

$$\ell = -\sum_{i=1}^n \log\left[\sum_{k=1}^K \pi_k (2\pi)^{-d/2} |S_k|^{-1/2} \exp[-\tfrac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top S_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)]\right], \tag{3}$$

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

2. (4 pts) Next, we apply (the adapted) Algorithm 1 in Ex 2.1 to the MNIST dataset (use the same configuration as in Exercise 1). For each of the 10 classes (digits), we can use its (and 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:

$$\hat{y}(\mathbf{x}) = \arg\max_{c=0,\ldots,9} \underbrace{\Pr(Y=c) \cdot p(X=\mathbf{x}|Y=c)}_{\propto \Pr(Y=c|X=\mathbf{x})}, \tag{4}$$

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 rates on the test set as a function of K. Use $5, 10, 20, 30$ as the values for K. Note that 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. Extra: 0.2 pts Do you think this will be true for harder classification problems (such as indoor scene classification)? Explain.]

3. (Extra: 2.5 pts) Redo Ex 2.1 where we fit d GMMs (each with K components) to each feature dimension $X_{:,j}$, separately. This is known as a component-wise GMM (cGMM). Implement the resulting EM algorithm. Analyze the space and time complexity of your implementation, and evaluate it on the MNIST dataset. Note that in this case, you do not need to perform classification, but rather apply cGMM to your entire dataset to cluster it into $K=10$ clusters.