Homework 5: Clustering & Mixture Models

UC Irvine CS275P: Machine Learning with Generative Models

Homework due at 11:59pm on May 20, 2025

Full Name:	**UCINetID (e.g., ucinetid@uci.edu):**

Question 0: (5 points)

Canvas includes detailed guidelines on how homework solutions should be formatted for submission to Gradescope. To receive full credit on this question, be sure that you read and follow these guidelines carefully! In particular, please take care that:

- There are two Gradescope assignments for Homework 5. For the first assignment, submit a PDF containing your answers for questions 1-3. For the second assignment, submit your Python code for question 3, which will be checked by an autograder. *For this assignment, you do not need to submit your code for question 2.*
- For the PDF submission, you must fill in this PDF template with your answers. All pages must remain in their original order when uploading your assignment to Gradescope. *Do not rearrange, add, or remove any pages from the provided PDF template.*
- For multiple-choice questions, ensure that you *completely fill in the bubble next to your selected answer*, and provide an explanation or justification to support your answer.
- Enter your final answers (to a precision of 3 decimal places) in the answer boxes, and write your explanations and/or math justifications in the space provided below each question. *Always show the work needed to produce your answers, not just the final result.* Math may be handwritten, but if possible, please type any sentences.
- For question 3, use the provided Python file. After adding your code to the template, submit that file separately to the second Gradescope assignment. In addition to uploading your Python code, be sure you provide answers and explanations in the designated areas of the PDF. *Do not include code for question 3 in the PDF.*
- If any question asks you to create a plot, insert an image showing the plot in the PDF.
- Check that your answers (especially scanned math) are readable within Gradescope, and that content has not been cut-off by the page margins.

Question 1: (30 points)

The four datasets illustrated in Figure 1 have two real-valued features x_1, x_2. For each dataset, we show a candidate separation into two clusters by marking points in one cluster with filled circles, and points in the other cluster with open squares.

Below, we ask questions about the behavior of the Expectation-Maximization (EM) algorithm for fitting a mixture of 2 Gaussians to each dataset:

$$p(x) = \sum_{k=1}^{2} \pi_k \mathrm{Norm}(x \mid \mu_k, \Sigma_k)$$

Assume a small amount of regularization is applied to the Gaussian variances, so that EM does *not* learn degenerate mixtures where the variance of one component shrinks to zero.

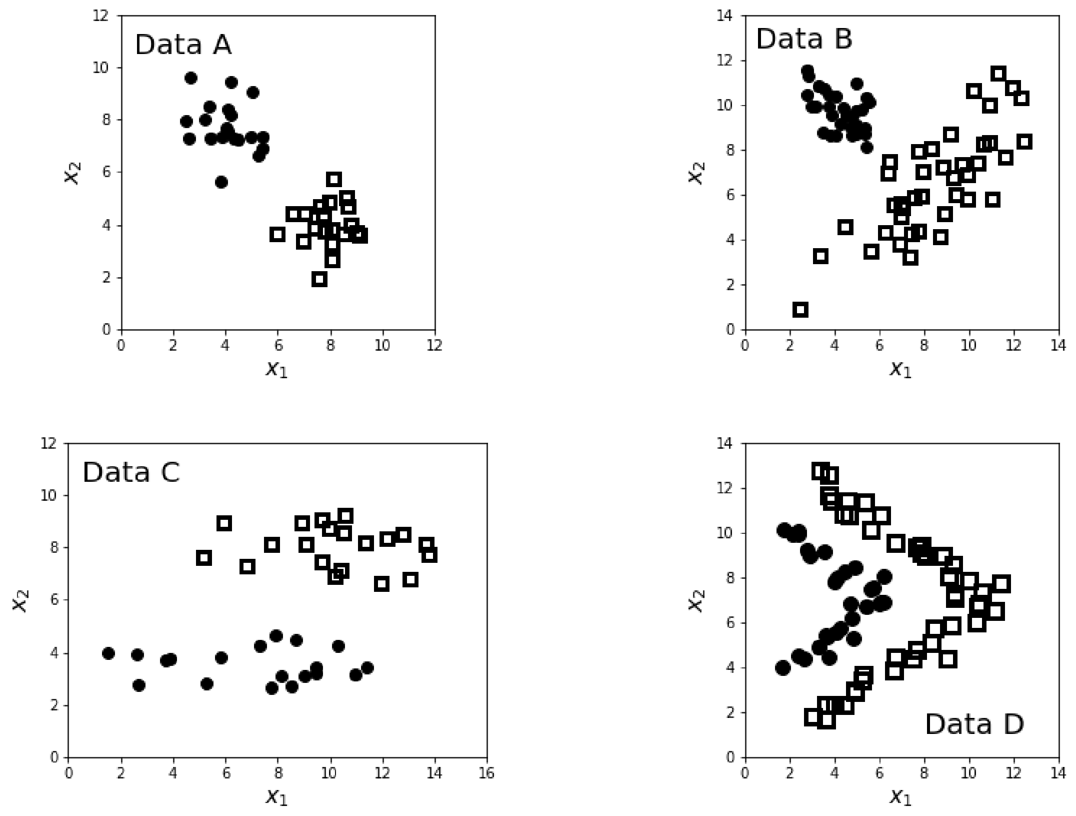

Figure 1: For each of these four datasets, we show a candidate separation into two clusters by marking points in one cluster with filled circles, and points in the other cluster with open squares.

a) *Suppose that the EM algorithm learns a spherical Gaussian mixture where the covariances $\Sigma_k = \sigma_k^2 I$ are multiples of the Identity matrix. (There is 1 variance parameter per cluster.) For each of the four datasets, would the given clusters have maximum (or close to maximum) data likelihood? Or would another clustering have substantially higher data likelihood? Briefly justify your answers.*

Data A: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data B: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data C: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data D: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Brief justification for answers:

b) *Suppose that the EM algorithm learns a diagonal Gaussian mixture where the covariances Σ_k are diagonal matrices. (There are 2 variance parameters per cluster, one per dimension.) For each of the four datasets, would the given clusters have maximum (or close to maximum) data likelihood? Or would another clustering have substantially higher data likelihood? Briefly justify your answers.*

Data A: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data B: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data C: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data D: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Brief justification for answers:

c) *Suppose that the EM algorithm learns a Gaussian mixture where the covariances Σ_k are full 2×2 matrices. (Because covariance matrices are symmetric, there are 3 covariance parameters per cluster.) For each of the four datasets, would the given clusters have maximum (or close to maximum) data likelihood? Or would another clustering have substantially higher data likelihood? Briefly justify your answers.*

Data A: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data B: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data C: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Data D: Do given clusters (approximately) maximize data likelihood?

◯ **Yes** ◯ **No**

Brief justification for answers:

Question 2: (30 points)

We will now use the EM algorithm to learn Gaussian mixture models of image data. We will cluster a collection of 16,688 images of natural scenes, taken from the larger SUN Database. To make the EM algorithm run more quickly, we used Principal Component Analysis (PCA) to reduce the dimension of each 32×32 color image to $D = 50$.

We will use the EM algorithm implementation from the `pomegranate` Python package, as provided by the `CustomGeneralMixtureModel` class. This class, which is distributed in `general_utils.py` in the code handout, extends the `pomegranate` `GeneralMixtureModel` class to add additional features. See the demonstration code for more details.

a) *Run the EM algorithm to learn a mixture of $K = 10$ Gaussians, where the covariances Σ_k are full $D \times D$ matrices. The provided support code shows how to use the "k-means++" initialization to reduce local optima. Use the* `plot_clustered_images` *function to plot some images with high probability of being generated by 10 of the learned clusters. Briefly discuss what aspects of the images seem to be captured by the learned clusters.*

Discussion of what aspects of images are captured by the learned clusters:

b) *Repeat part (a) with a mixture of $K = 20$ Gaussians. Again plot some images with high probability of being generated by the learned clusters. Which of the two models seems to provide a higher-quality clustering of the images?*

Discussion of which model provides a higher-quality clustering of the images:

c) *Using the mixture of $K = 20$ Gaussians from part (b), use the* **`plot_sampled_images`** *function to generate and plot synthetic images using 10 of the learned Gaussian clusters. Briefly discuss some aspects of the real images that are captured by the multivariate Gaussian clusters, and some aspects that are not well modeled.*

Discussion of how well images are captured by the Gaussian cluster shapes:

Question 3: (20 points)

In this question, we will implement a Bernoulli mixture model to perform unsupervised clustering on a dataset of English words annotated by a large set of binary features. There are a total of 541 unique words labeled with 824 binary features describing properties of each word, such as "is a musical instrument" or "has buttons".

We will use the EM algorithm implementation from the `pomegranate` Python package, as provided by the `CustomGeneralMixtureModel` class. This class, which is distributed in `general_utils.py` in the code handout, extends the `pomegranate GeneralMixtureModel` class to add additional features. You should also use the `FixedBernoulli` class from `word_utils.py` in the code handout, rather than the standard `pomegranate Bernoulli` class, to avoid numerical errors. See the demonstration code for more details.

a) *Run the EM algorithm on the word feature dataset, using $K = 8$ clusters and the options specified in the demonstration code. Plot the log-likelihood versus iteration. If you ran the EM algorithm on a different dataset of binary features, would the log-likelihood be guaranteed to monotonically increase?*

Is EM guaranteed to monotonically increase the log-likelihood on other datasets?
◯ ***Yes*** ◯ ***No***

b) *Repeat this experiment 10 times, running EM from 10 random initializations. Select the models with the highest and lowest final log-likelihoods. For these two models, use the* ***predict_log_proba*** *method from* ***CustomGeneralMixtureModel*** *to calculate the posterior distributions for assigning each word to the various mixture components. List the 5 words most likely to be associated each of the $K = 8$ mixture components, for each model. Do they correspond to meaningful groupings or categorizations? How do the best and worst runs differ in terms of their word clusterings?*

5 most likely words per cluster for the highest and lowest likelihood models:

Brief discussion of results:

c) *Use the feature labels included in the dataset to list the top five features associated with each mixture component, by sorting the probabilities of each mixture component distribution from the learned mixture model. You may retrieve the mixture component distributions from the* `distributions` *variable for the learned mixture model, and then use* `get_p_values_numpy()` *to get the corresponding Bernoulli likelihood parameters. Compare the top five feature labels for the best and worst runs. Is there a significant difference in the coherence or interpretability of these feature labels?*

5 most likely features per cluster for the highest and lowest likelihood models:

Brief discussion of results: