BERNARD BARUCH COLLEGE

MTH 4330 INTRO TO ML (SPRING 2023)

TAKE HOME FINAL

This Final is about the use of Gaussian Mixture Models (GMM) for a series of different purposes. There are no right or wrong anwers, I will pay particular attention in the way you justify your procedures and the clarity you use to describe them. The output should not have more than 10 pages on a pdf file.

The project is divided in 3 parts. In the first part you will use GMM as a density estimator algorithm. In the second part you will use the label in the data set to refine your GMM as a classification algorithm. In the third part you will work on the MNIST dataset and have to come up with your own way of improving the GMM by means of a different classifier as for instance decision trees.

GMM as a density estimator algorithm. In the first part of the project we will use a Gaussian Mixture Model (GMM) to perform density estimation on the two-moon example.

- Download the data set relative to two clusters of points displayed as two moons.
- Build a two component GMM and plot the result visualizing the data and the level sets of the two Gaussians (plot of two different colors the level sets relative to the two different Gaussians).
- What do you observe? If we were to use Gaussian mixture as a clustering algorithm (each point is assigned to the component that has the largest value when evaluated at that point), would you say that a two components GMM is able to capture well the two clusters?
- If you are thinking that a solution is to increase the number of components, how would you choose such number? There are many ways to do so, one, for instance is to use the BIC criterion on a test set, you can look here as a starting point for its use with GMM.The number of components should be something in between 10 and 15. As usual, after you decided how many number of components you need and explained the procedure for selecting that number, you should plot the level sets of your GMM and check whether they are compatible with the sample points.

GMM as a classifier. In this part we will use the labels of the data relative to the two-moon example.

- Plot the data and, if the data set doesn't come already with labels, assign to each point of the data set either the label $y = 1$ if the point belongs to the upper moon or $y = -1$ if the point belongs to the lower moon (color the points with two different colors depending on the cluster they belong to).
- You can use a GMM to perform classification using two components, one for each class. After your trained your GMM each point can be assigned to a class by looking at $P(X|Y = i)$ where i can be ± 1. However, it is clear that with only two components the GMM will perform poorly as a classifier. Plot the points coloring them according to $P(Y|X = x)$ as we explained in class.
- How can we use your GMM to perform classification when you use more components than classes? There are many ways to do so and you can come up with your own (describe it clearly though). A starting point could be the following:
  - Assign each component of your GMM to a class. You can do this in many ways; one starting point is to evaluate each component at the points of each class separately and define a metric (for instance according to a majority rule: within one standard deviation of the component, are there more points with $y = 1$ or $y = -1$? in the first case assign the component to the class $y = 1$ or to $y = -1$ otherwise) to identify to which class that component belongs.
  - Given a new observation x (a new point for which we don't know the label), you can decide to which class it belongs by evaluating each component of your GMM at that point (computing $P(Y = i|X = x)$ for each component i) and assigning the point to the component corresponding to the largest probability.
  - Assign that point to the class that component belongs to.
- Plot the points together with the level sets of the GMM components and color the points according to what class is the GMM assigning them. What do you observe?
- If needed you can increase the number of components of your GMM by minimizing the classification error. You can do so by dividing the data set in a training set and a test set and increase the number of components until the error (you can come up with your own metric or get some inspiration from the

concept of F1 error or weighted empirical error for classification; in any case, justify your choice) on the test set is at its minimum. Remeber not to overfit the data.
- What is the difference with training your GMM without looking at the label and just considering it as a density estimator algorithm?
- The nice part of using a GMM is that now, once you believe you selected the right number of components in your model, you can *generate* points from a GMM. The reason is that it is relatively simple to generate points from a Gaussian distribution (see the command numpy.random.normal) and therefore from a GMM. Generate 500 points from your GMM, does it seems to you they are reproducing the two moons?

GMM on the MNIST dataset as a generative model. Do for the MNIST data set what you did for the two-moon example. In this data set each sample point (representing a hand written digit, represents a figure) is defined in a high dimensional space (whose dimension is given by the number of pixels used to represent the figure). It is therefore advisable to reduce the dimensionality of the space using PCA. I believe that the number of components should be anything between 30 and 100, you should justify the number you pick by experiments or by defining (and justfying) a metric that you believe is relevant for the problem.

The overall goal is to build a GMM that can generate new handwritten digits:

- Perform PCA on the original dataset. Justify the number of components you are choosing.
- Build a GMM as you did in part one of this final. Train first a GMM without looking at the labels (considering it as a density estimator) and then using the labels. Choose what you believe is the best in between the two models you obtain.
- For each class of the dataset (that is characterized by 10 classes each one representing a digit), find the GMM components belonging to that class and plot their means. If you did things correctly, each mean should be identifiable as a digit.
- Sample the GMM and plot some of these samples, do they look like hand written digits?
- Another way to check whether your GMM is performing well as a classifier is to compare its performance by using a different classifier, like, for instance, a boosted gradient tree. If you generate points from the GMM and assign them the label based on which component the point is coming from (remember that each component is assigned to a given class), does your tree classify that point correctly?