ASSIGNMENT 3

Mathematics for Machine Learning 811 31 January 2022

All of these problems can be done in Python. When asked to implement algorithms like PCA or EM, the idea is that you do so "from scratch" (following the steps in the lecture slides), instead of calling an existing function.

What you submit must be your own work, and sources other than the lecture material must be cited. Remember to append a signed plagiarism declaration to your submission.

---

1. For this problem we consider the famous MNIST dataset; a large collection of handwritten digits (0 to 9) in the form of 28×28 greyscale images (60,000 for training and 10,000 for testing).

(a) Find, download and import the MNIST dataset into your coding environment, and display 4 randomly selected images of each digit in a 4×10 grid.

(b) Consider the set of all training images of the digits 0 and 1. Reshape each image into a 784-dimensional vector, and perform PCA in order to project these vectors to 2 dimensions. Plot the 2-dimensional embeddings as dots, using different colours for 0's and 1's, similar to Figure 10.10 in the textbook.

(c) Now perform PCA on the set of all training images of the digit 8, again reducing the dimensionality to 2, and plot the embeddings of this set as dots. Generate new images of the digit 8 by decoding any vector $\boldsymbol{z}$ in the 2-dimensional space, reshaping the decoded vector to 28×28 and displaying it as an image. Experiment with choosing $\boldsymbol{z}$ close to *vs* away from the embeddings of the training set, similar to Figure 10.15 in the textbook.

2. Consider the small set of face images accompanying this assignment, split into training and test data. Perform PCA on the training set (again, reshape all the images to column vectors) in order to reduce the dimensionality of the images to M. Pick two images from the training set and two from the test set at random. Encode and then decode (reconstruct) each one, and compare to the original images. Do all of this for $M = 1, 5, 10, 20, 40$.

Note: you will perform PCA 5 times, and generate 5 reconstructions of each of the 4 images you selected.

3. A simple algorithm for sampling from a given GMM is first to select a mixture component at random (using the mixture weights as a distribution for this selection) and then to draw a sample from that component (using the probability integral transform implemented in Assignment 2, for example).

(a) Put this algorithm to the test, by creating a small 2-dimensional GMM with say 4 components (choose means, covariances and mixture weights yourself), sampling say 1,000 points from that GMM, and then running the expectation maximisation algorithm on those samples to recover the GMM's parameters.

(b) Visualise the responsibilities of all the data points in a 2D plot (see Figure 11.10b) through the course of part (b)'s EM iterations: at initialisation, at some point roughly midway through, and at the end (at convergence).

4. *see next page...*

4. Train linear one-vs-rest SVMs on the MNIST training set. The inputs should be 784-dimensional vector versions of the images. Here you may use `sklearn.svm.linearSVC`, but you must train a collection of **binary** classifiers and then implement a suitable combination of their outputs on test data yourself.

Experiment with the regularisation parameter C, and investigate its effect on test accuracy. Finally, show a handful of incorrectly classified test images with their predicted and expected labels, and discuss.

Note 1: strictly speaking, we should never tune a hyperparamter like C on test data (why not?). For the purposes of this problem, let's think of the MNIST test data as our validation set.

Note 2: if you experience computational or memory limitations with the full MNIST training set, you are welcome to consider a smaller, randomly selected subset (but be sure to keep the label distribution more-or-less uniform).
