HW 3

Due: Tuesday, April 11, 2023 at 2:00 pm.

Please hand in this Homework as follows:
- Upload to Gradescope `HW3`: A pdf of problems 1,4 **combined with**.
- Upload to Gradescope `HW3 programming`: A `ipynb` file for problems 2,3.

1. *Bernoulli models:* (15 points)

   (a) Assume your observations are binary in R^d, i.e. $X = (X_1, \dots, X_d), X_j \in \{0, 1\}, j = 1, \dots, d$. We assume the d variables are independent with joint distribution $P(X_1 = x_1, \dots, X_d = x_d) = P(X_1 = x_1) \cdots P(X_d = x_d)$, and $P(X_j = x) = p_j^x(1 - p_j)^{1-x}$. Given a sample $X_1, \dots, X_n$ from this distribution write the log-likelihood, write the score equation for each parameter $p_j, j = 1, \dots, d$ and write the maximum likelihood estimate $\widehat{p}_j$. (The estimates should be very simple and intuitive...)

   (b) Assume you have a Bernoulli model for each of C classes, with probabilities $P(X_j = 1|Y = c) = p_{c,j}, c = 1, \dots, C, j = 1, \dots, d$ and $\pi_c = P(Y = c)$. Write out the Bayes classifier in terms of these models, show that it is a linear classifier $\arg\max_c h_c(x)$, with $h_c(x) = W_c x + b_c$. Write out W_c and b_c in terms of the parameters of the model.

   (c) Assume now that the data $X_1, \dots, X_n$ come from a mixture model with M components and each component a product of d independent Bernoulli variables as in (a):

$$f(x; \theta) = \sum_{m=1}^{M} \pi_m f_m(x; \theta_m), \quad f_m(x; \theta_m) = \prod_{j=1}^{d} p_{j,m}^{x_j}(1 - p_{j,m})^{(1-x_j)},$$

   where $\theta = (\pi_1, \dots, \pi_M, \theta_1, \dots, \theta_M)$ and $\theta_m = (p_{1,m}, \dots, p_{d,m})$.
   We will derive the details of the EM algorithm for this model.

      i. Write out the precise formula for computing the responsibilities

$$w_{mi} = P(m|X_i; \theta), m = 1, \dots, M, i = 1, \dots, n$$

      in terms of the current estimates $\widehat{\pi}_m, \theta_m, m = 1, \dots, M$.

      ii. In computing the responsibilities you would be computing ratios involving the f_m's. If d is large you are multiplying lots of small numbers on a computer and quickly reaching the rounding error of the computer. You may be dividing by very small numbers, which is risky on the computer. Instead it is convenient to write:

$$f_m(X_i; \theta_m) = \exp[\sum_j X_{ij} \log p_{j,m} + (1 - X_{ij}) \log(1 - p_{j,m})].$$

For each data point, how would you use this expression for the different clusters to guarantee that the ratio you compute always has a value less than 1 in the numerator and a value greater than 1 and less than M in the denominator.

iii. Once you have the responsibilities. Write the formula for computing the new estimates $\widehat{\pi}_m^{new}, p_{j,m}^{new}$. Remember, all data points contribute to the estimates of each of the cluster parameters, and their contribution is weighted by the w_{mi}. So you should get a weighted version of the estimates in 1a.

2. *Neural Data* (30 points) The file linked here is a voltage recording from one electrode in the brain of a marmoset over the period of about 30 minutes. You can also access the file in the `Neuro` subfolder of the data page in Canvas under the name: `FilteredRecordings.npy`. The data comes from Jeff Walker working in the lab of Prof. Nicho Hatsopoulos. The data X is obtained at 30Khz, i.e. 30,000 observations per second and we have already filtered its spectrum to lie more or less between 300Hz and 3000Hz.

   (a) Threshold the data at 95 and find the contiguous segments of data above that level. For each initial time point t of a contiguous segment extract a segment of 80 time points $X(t-40 : t+40)$. You should get about $n \sim 18,000$ length 80 curves. Plot some of them to get a sense of your data. Arrange these curves in an $n \times 80$ array Z.

   (b) We'd like to cluster these curves but the problem is that they are not aligned. One way to align them is to match their maxima. For each curve $Z[i,:]$ find its time t_i of maximal value and extract a shorter curve of length 50 **centered** at t_i. (If t_i is too close to the beginning or the end just drop that sample.) This will yield a new array Y of dimension $m \times 50$, with m a bit smaller than n. All curves in Y have their maximum at time point 25.

   (c) Now perform Kmeans clustering on these curves, with 2,3 and 4 clusters. Plot the cluster means and the curves assigned to each cluster.

   (d) A different way to cluster is to first perform dimensionality reduction using PCA on the first 2 principal components. Write your own PCA routine (don't use the one supplied by python). Of course you can use the python or scipy eigen function. Plot the projection of the data onto the first two principal components. (Use very small dots in order to better see the structure of the data.)

   (e) Now cluster the data using Kmeans using the 2d projections into 2,3, and 4 clusters. Plot the data with their cluster assignments in color.

   (f) Use the corresponding cluster assignments to produce mean curves for each cluster. Compare these mean curves to the ones you got clustering the curves directly.

3. *Handwritten digits* (40 points)

   For this extended problem you are going to work with the MNIST handwritten digits dataset. The dataset is made up of more than 70,000 images of the digits 0-9. The data are length 784 vectors ranging between [0,255] representing intensity values. The vectors are a flattened version of the 28×28 pixel scans of the digits. The data is in the `mnist` folder of the course `Data folder`. Its called `MNIST_data.npy` and `MNIST_labels.npy` and can be loaded with the `np.load` command. After you have read in the data, you should divide the data by 255 so that the values lie in $[0, 1]$.

   To display the images you can do something like this: store a few digits into a numpy array `examples`, whose rows are digits.

```
%matplotlib inline
import matplotlib.pyplot as plt

nrows = 4;ncols = 5
plt.figure(figsize=(ncols*2, nrows*2))

for i in xrange(nrows*ncols):
    plt.subplot(nrows, ncols, i+1)
    plt.imshow(examples[i].reshape((28,28)), cmap='gray')
    plt.axis('off')

plt.show()
```

   (a) *Extract principal components.*

   Using the code you wrote in the previous problem PCA to extract the principal components of the training data. Display the first 10 principal components as images.

   (b) *Plot variance*

   Plot the variance of all of the principal components—this corresponds to the eigenvalues. This should be monotonically decreasing.

   (c) *Dimension reduction*

   Take a data point in the test data set and project it onto the first m principal components (remember to first subtract the mean). Then, transform that m-length vector back into a 784-length vector and display it as an image (add back the mean). Repeat this for several different values of m. Also, try it on different data points. Describe the results qualitatively. Does it give an accurate representation of the images? How do the results depend on m? Can you describe what the top principal components are capturing?

   (d) *k-means*

   Now use k-means to perform unsupervised clustering of the digit data, and try to assess how well the clusters capture the structure of the data. Do not produce more than 50 clusters. Show the centers of the clusters. To evaluate how closely the clusters capture the structure of the data, associate each cluster with the majority label.

Construct plots showing samples of the digits from each cluster, and comment on how well the clusters respect the true digit labels.

(e) *Bernoulli mixtures*

Binarize the dataset by thresholding each pixel value at $\tau = .5$. Implement the EM algorithm from the previous problem with $M = 10, 20$

Some important things to keep in mind:

- Make sure you take advantage of matrix- vector and matrix-matrix multiplications. Many of the operations you wrote down for the algorithm can be written in this form.
- To initialize the algorithm you can simply start with a random $N \times M$ positive matrix w with rows that add up to 1.
- You want to avoid probabilities that are 0 or 1, so every time you estimate a binary probability as a ratio a/b, modify it to $(a+1)/(b+2)$.
- Sum the log-denominators of the responsibilities $w_{m,i}$ over all examples:

$$L = \sum_{i=1}^{n} \log \sum_{m'=1}^{M} f_{m'}(X_i, \widehat{\theta}_{m'})\pi(m').$$

  This gives the current log likelihood of the data. This quantity should decrease with each iteration. One criterion to stop the iterations is when this quantity doesn't change much.
- Each component of the mixture model is defined by a vector of probabilities $p_{m,j}, j = 1, \ldots, d$, with $d = 784$. You can reshape this array into a 28×28 image and show it. You should be observing smooth images of some of the digits.

4. *Bayes rule, variances, and priors.* (15 points) Let $f(x|Y = k) \sim N(\mu_k, \sigma^2), k = 1, 2$ be a two category one-dimensional problem with $P(Y = 1) = \pi_1, P(Y = 2) = \pi_2$.

(a) Compute the decision boundary assuming $\mu_1 > \mu_2$ and compute the Bayes loss (=Bayes error).

(b) Write the probability of error using the CDF of the normal distribution. Show that the error goes to zero as $\sigma \to 0$.

(c) For fixed σ what happens to the decision boundary as $\pi_1 \to 0$? For small π_1 what is a very simple classification rule that can guarantee low error rate?

(d) Assume a classification problem with K classes and $x \in R^d$. Let $L(k,l)$ be the cost of choosing class l when class k is true. Let $p(x,y)$ be the joint distribution of X, Y. The expected loss for a decision function h is

$$\mathcal{L}(h) = E\left[L(Y, h(X))\right] = \sum_{k=1}^{K} \int_{R^d} \sum_{l=1}^{K} \mathbf{1}[h(x) = l] p(x,k) L(k,l) dx.$$

Show that the decision rule with lowest loss is given by the generalize Bayes rule:

$$h_B(x) = \text{argmin}_{j=1,\ldots,K} \sum_{k=1}^{K} L(k,j) p(k|x).$$

Hint: Use the loss of an individual example defined as $\mathcal{L}(h,x) = \sum_{k=1}^{K} L(k, h(x)) p(k|x)$.

(e) One way to avoid the degenerate situation from item 4c is to change the cost function. Set $L(2,1)$ to be the cost of choosing class 1 when class 2 is true, and $L(1,2)$ the cost of choosing class 2 when class 1 is true. Write the expected loss in this situation. Recompute the decision boundary. What values on $L(i,j)$ would you assign to remedy the problem of small π_1.