1. Problem 3.8.38 in DHS

2. Problem 4.5.8 in DHS

3. BDR and nearest neighbors Consider a classification problem with $c$ classes and uniform class probabilities, i.e. $P_Y(i) = 1/c, \forall i$. Assume that the goal is to classify an iid sequence of observations $X = \{x_1, \ldots, x_n\}$ as a whole (i.e. the samples are not classified one at a time).

a) Compute the BDR for this problem and show that it converges (in probability) to a nearest neighbor rule based on the class-conditional distributions and the distribution of the observations. Show that the distance function is the Kullback-Leibler divergence

$$D[p(x)||q(x)] = \int p(x) \log \frac{p(x)}{q(x)} dx.$$  

This proves that the BDR for the classification of sequences is really just a nearest neighbor rule.

b) Assuming that all densities are Gaussian with equal covariance $\Sigma$, the class conditional densities have mean $\mu_i$ and the observation density has mean $\mu$ write down an expression for the decision rule as a function of the Gaussian parameters. Provide an interpretation for this new decision rule, by stating what are the items being compared and what is the distance function.

4. Kernel methods Problem 4.3.2 in DHS (do a) and b) only)

5. Kernel methods Problem 4.3.3 in DHS

6. (Computer) This week we use the cheetah image to compare the various classifiers that we have discussed in class. Once again we use the decomposition into $8 \times 8$ image blocks, compute the DCT of each block, and zig-zag scan. We then compute the PCA of the scanned vectors and compare four classifiers

1. the BDR in DCT space assuming Gaussian classes
2. the BDR in PCA space assuming Gaussian classes
3. the nearest-neighbor classifier in PCA space under the Euclidean distance
4. the BDR in PCA space with densities estimated non-parametrically with a Gaussian kernel of identity covariance,
trained with the data in `TrainingSamplesDCT_new_8.mat`.

**a)** Classify the *cheetah* image with each of the classifiers, using the following numbers of dimensions \{1, 2, 4, 8, 16, 24, 32, 48, 64\}. For each classifier, plot a curve of the classification rate as a function of the number of dimensions. Comment the results addressing at least the following three questions: 1) does PCA have any advantage over the DCT?, 2) how do the different classifiers perform relative to each other?, 3) what is the behavior in terms of the number of dimensions? Explain why. Finally, comment on any other unexpected results that these plots may suggest.

**b)** Repeat the experiment above for the classifier based on the Gaussian kernel, for various values of the covariance matrix obtained as follows: i) Start from the diagonal covariance matrix resulting from the PCA analysis; ii) multiply by a scalar ranging from 0.1 to 2 with intervals of 0.2. For each kernel, plot a curve of classification rate as a function of dimension, using the set of dimensions introduced in **a**). Comment on the impact of the choice of covariance.

**c)** Compare the first PCA component with the Fisher linear discriminant in two ways. i) classify the *cheetah* image with both for the set of dimensions above and plot the resulting error rate. ii) compute, and plot, the angle between the two vectors as function of the dimension. Comment on the results.

**Important note:** How you write your code makes a tremendous difference on how long the problem will take to compute. Recall that the NN classifier requires the computation of a large number of distances. To make these feasible avoid loops at all cost. Here is a Matlab trick that will help. If the vectors \(x_i\) are the rows of matrix \(X\), the evaluation of the distance from a new row vector \(x\) to all \(x_i\) can be performed efficiently through something like

\[
M = \text{ones} (\text{size}(X, 1), 1) * x - X^2
\]

Note that, by using the `cumsum()` on the rows of \(M\) and then adding along the columns you immediately get the distances for all numbers of dimensions. To evaluate the kernel density estimate you can use similar tricks.