**Amirkabir University
of Technology**
(Tehran Polytechnic)

Assignment 4
From *Estimating Densities* and *Reducing Dimensionalities* to *Classifying Samples*

Please note:

1. What you must hand in includes the assignment report (.pdf) and – if necessary – source codes (.m). Please zip them all together into an archive file named according to the following template: HW4_XXXXXXXX.zip
   Where XXXXXXXX must be replaced with your student ID.

2. Some problems are required to be solved *by hand* (shown by the ✎ icon), and some need to be implemented (shown by the 🔺 icon).

3. As for the first type of the problems, you are free to solve them on a paper and include the picture of it in your report. Here, cleanness and readability are of high importance.

4. Your work will be evaluated mostly by the quality of your report. Don't forget to explain what you have done, and provide enough discussions when it's needed.

5. 5 points of each homework belongs to compactness, expressiveness and neatness of your report and codes.

6. By default, we assume you implement your codes in MATLAB. If you're using Python, you have to use equivalent functions when it is asked to use specific MATLAB functions.

7. Your codes must be separated for each question, and for each part. For example, you have to create a separate .m file for part b. of question 3. Please name it like p3b.m.

8. Problems with bonus points are marked by the ⭐ icon.

9. **Please upload your work in Moodle, before the end of May 25nd.**

10. If there is *any* question, please don't hesitate to contact us through the following email address:

    - **b.roshanfekr@yahoo.com**

11. Unfortunately, it is quite easy to detect copy-pasted or even structurally similar works, no matter being copied from another student or internet sources. Try to send us your own work, without being worried about the grade! ;)

1. Basic Density Estimation with Non-Parametric Methods (12 Pts.)

Keywords: *Density Estimation, Non-Parametric Methods, Histogram, Parzen Windows, Kernel Density Estimation, K-Nearest Neighbors*

Density Estimation is the problem of reconstructing an estimate of the probability density function using a set of observed data points. In other words, we observe $X_1, X_2, ..., X_n$ and we would like to recover the underlying probability density function generating this dataset. While a classical approach of density estimation is the **Histogram**, we often seek more complicated methods like **Parzen Windows** or **Kernel Density Estimation**.

Let's practice these methods on some routine density estimation problems. First, consider a set of observed data $D = \{0,0,1,1,1,1,2,2,3,4,4,5,5,5\}$.

a. Draw a histogram of the data, with the bins centred at $\{0,1,2,3,4,5\}$ with the width of 1.
b. Sketch the Parzen window estimate of the unknown density function for $h_n = 1,3,5$.
c. Sketch the 2-nearest neighbor estimate of the density function.
d. Consider the following triangle kernel as the window function

$$K(u) = (1 - |u|)\delta(|u| \leq 1)$$

where $u = x - x_i/h$. Find the kernel density estimates for $x = \{0,1,2,3,4,5\}$ and bandwidths of 3.

In a slightly different scenario, assume $f(x)$ is an unknown density function and the following samples are draw from $f(x)$:

$$X = \{4,5,5,5,6,6,9,11,11,11,15,16,16\}$$

e. Estimate the density $p(x)$ at $y = 3,10,15$, assuming a bandwidth of $h = 4$ and standard kernel function

$$K(u) = \begin{cases} 1 & |u| < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

f. Repeat the previous part using a Gaussian kernel $N(0,1)$.
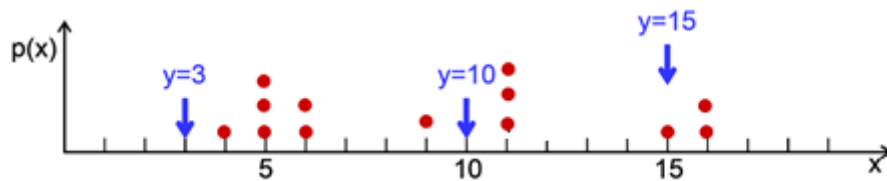

Figure 1 The distribution of the given dataset in part e. and f.

Finally, assume two Gaussian distribution with $N(-1.5, 2.5)$ and $N(1.5, 1.75)$. Generate 100 samples from each distribution and label them as ω_1 and ω_2.

g. Plot a Parzen window estimated likelihood $\hat{p}(x|\omega_1)$ and $\hat{p}(x|\omega_2)$. For the window, use a Gaussian window with the size of h, i.e. $\phi(x) = e^{-x^2/h}$.
h. Plot the K-NN estimated likelihood $\hat{p}(x|\omega_1)$ and $\hat{p}(x|\omega_2)$.
i. Assuming $p(\omega_1) = p(\omega_2)$, compare classification accuracy of parts g. and h.

2. Evaluating K-NN in Simple Classification (and Maybe Regression!) Problems   (14+6 Pts.)

Keywords: *Classification Problem, Regression Problem, K-Nearest Neighbors, Cross Validation, Leave-One-Out*

Density Estimation techniques can be used to perform classification. In fact, using **Kernel Density Estimator**, one can estimate the joint density $P(Y, X)$, then use it to calculate $P(Y \mid X)$. This is where the **K-Nearest Neighbors** algorithm comes in; a simple yet effective classification method which uses a distance metric in order to determine the k samples that are closest to the test sample, then classifies it by a **Majority Vote** of its neighbors. It is usually applied with a **Model Validation** technique, such as **Leave-One-Out**, to justify how accurately it performs in practice.

Let's start with an easy problem. Given the dataset bellow

X	−3.1	−2.31	−1.75	−1.3	−0.6	0.2	0.4	1.3	1.4	2.0	2.5	2.7
label	−	+	+	+	+	−	+	−	+	−	−	−

  a. Calculate the leave-one-out cross validation error of 1-nearest neighbor.
  b. Calculate the leave-one-out cross validation error of 3-nearest neighbor.
  c. Which one would you choose for this dataset: 1-NN or 3-NN?

In another scenario, there are two classes, such that

$$\text{samples from class 1: } \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{samples from class 2: } \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

  d. Draw the 1-NN decision boundaries for the given set of sample points.
  e. Draw the 2-NN decision boundaries. Remember to indicate the reject region.

Now let's apply k-nearest neighbors to another binary classification task shown in Figure 2.

  f. Find the value of k which minimises leave-one-out cross-validation error.
  g. What is the resulting error?
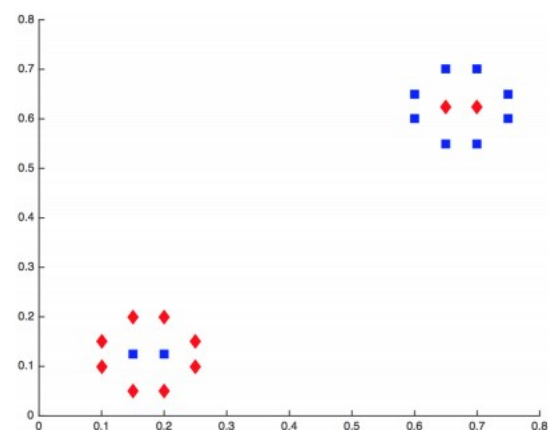
Note: A point can be its own neighbor.

Figure 2 Data samples of part f. and g. with class 'blue' and class 'red'

In a different dataset, consider a binary classification task illustrated in Figure 3.

h. Find the value of k which minimises the training set error for the given dataset.

i. Compute the resulting training error.

j. Why would selecting large values for k be bad in this dataset?

k. Why would selecting small values for k be also bad in this problem?
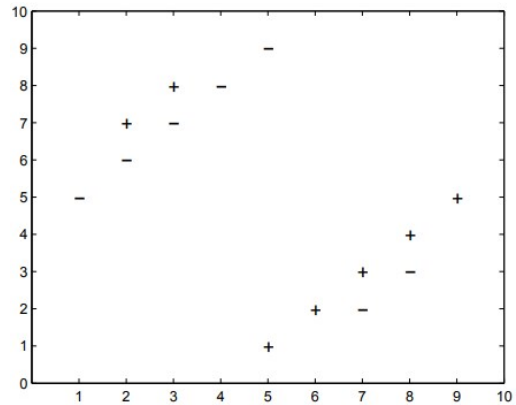
l. Draw the 1-NN decision boundary for this problem.

Note: You can use an image editor software to draw the required decision boundary on the image "p2_samples.jpg", which can be found in folder "inputs/P2".

Figure 3 Dataset given for part h. to part l., with two classes '+' and '-'

Up until now, you have applied k-nearest neighbors in some classification task. Now, let's get familiar with the application of K-NN in a different problem, i.e. **K-NN Regression**.

Consider a regression problem for predicting the data points given in Figure 4. The goal is to use k-nearest neighbors regression method to minimises mean squared error (MSE).

⭐ m. Find the training error when $k = 1$.

⭐ n. Find the leave-one-out cross validation error when $k = 1$.

⭐ o. Find the training error when $k = 3$.

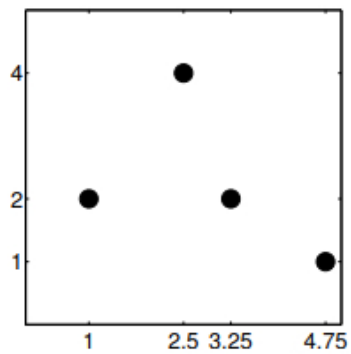⭐ p. Find the leave-one-out cross validation error when $k = 3$.

Note: By default, use Euclidean distance as distance metric.

Figure 4 Data points given for K-NN regression problem

3. Principal Component Analysis for Dimensionality Reduction   (8 Pts.)

Keywords: *Dimensionality Reduction, Principal Component Analysis (PCA)*

Dimensionality Reduction is the usage of a series of techniques to reduce the number of random variables to consider. It usually makes data analysing much easier and faster for machine learning algorithms, and helps avoiding many problems of high dimensional data, including **the Curse of Dimensionality**.

The most known linear technique in this area is **Principal Component Analysis (PCA)**, where the goal is to apply a linear mapping of the data to a lower dimensional space in such a way that the variance of the data in the lower dimensional representation is maximized.

Let X be a dataset such that

$$X = \begin{bmatrix} 1 & 0 & 7 & 4 & 1 & 2 & 10 & 2 & 2 & 12 \\ 5 & 5 & 9 & 9 & 6 & 6 & 11 & 6 & 6 & 9 \\ 8 & 6 & 12 & 9 & 4 & 9 & 13 & 8 & 10 & 6 \end{bmatrix}$$

a. Find the samples mean.
b. Centre the samples to have zero mean.
c. Find the covariance matrix of the zero-mean samples. How can you describe the given samples by inspecting the obtained covariance matrix?
d. Find the Eigenvalues and Eigenvectors of the computed covariance matrix.
e. What would be the new basis of the given data?
f. Project the samples on the new basis.

Now assume $(-1,-1)$, $(0,0)$ and $(1,1)$ are three points in the 2D space.

g. Write down the actual vector of the first principal component direction.
h. Project the data points onto the 1D space using the principal component computed in the previous part, and find the variance of the projected data.
i. Reconstruct the original data points in 2D space using projected points in the previous part, and find the reconstruction error.

4. Linear Discriminant Analysis: When Labels Matter (5 Pts.)

Keywords: *Dimensionality Reduction, Linear Discriminant Analysis (LDA)*

Another well-known technique for **Dimensionality Reduction** is **Linear Discriminant Analysis** (**LDA**), in which the main goal is to model the difference between the classes of data. In other words, **LDA** tries to project a dataset onto a lower-dimensional space with good class-separability in order to avoid overfitting (once again, **the Curse of Dimensionality**) as well as reduce computational costs. Therefore, **PCA** is considered as an unsupervised method, whereas **LDA** is somehow a supervised technique.

Assume two classes ω_1 and ω_2, where

$$\text{samples belonging to } \omega_1 : \begin{bmatrix} 0 & 6 & 4 & 4 & 3 & 2 & 1 & 2 & 3 & 5 \\ 3 & 10 & 4 & 7 & 8 & 5 & 3 & 4 & 6 & 10 \\ 9 & 12 & 1 & 8 & 9 & 10 & 4 & 10 & 10 & 8 \end{bmatrix}$$

$$\text{samples belonging to } \omega_2 : \begin{bmatrix} 6 & 4 & 8 & 2 & 3 & 3 & 4 & 9 & 6 & 0 \\ 4 & 12 & 6 & 5 & 5 & 6 & 9 & 9 & 10 & 2 \\ 5 & 12 & 6 & 15 & 3 & 5 & 6 & 11 & 6 & 9 \end{bmatrix}$$

a. Find classes mean.
b. Find the covariance matrix of both classes.
c. Compute the within-class scatter matrix.
d. Compute the between-class scatter matrix.
e. Find the LDA projection.
f. Project the given samples of both classes onto the computed LDA projection.

5. PCA vs. LDA: Comparison of Two Different Linear Projections   (16 Pts.)

Keywords: *Dimensionality Reduction, Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA)*

After practicing two popular methods in **Dimensionality Reduction**, i.e. **PCA** and **LDA**, it's time to take a closer look at their differences.

First, assume a 3-category dataset with the samples given in Figure 5.

a. Draw the first principal component direction considering samples of class 1 and class 2.
b. Draw the first Fisher's linear discriminant direction considering samples of class 1 and class 2.
c. Repeat part a. considering samples of class 1 and class 3.
d. Repeat part b. considering samples of class 1 and class 3.
e. Repeat part a. considering all samples.
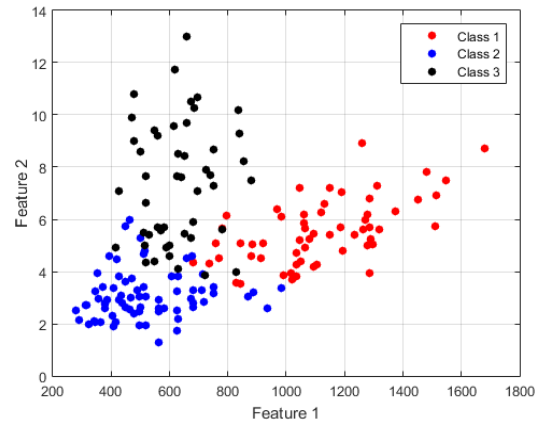f. Repeat part b. considering all samples.

Figure 5 Distribution of the samples in problems of part a. to part f.

Note: In order to draw the required lines, you can use image "pca_lda_sandbox.png" in "inputs/P5" folder and an image editor software like Photoshop or Paint.

Now, consider three densities with the following characteristics:

$$\mu_1 = \begin{bmatrix} 4 & 0 & 4 \end{bmatrix}^T, \quad \mu_2 = \begin{bmatrix} 5 & 6 & 7 \end{bmatrix}^T, \quad \mu_3 = \begin{bmatrix} 6 & 1 & 5 \end{bmatrix}^T$$

$$\Sigma_1 = \begin{bmatrix} 5 & -3 & -2 \\ -3 & 4 & 0 \\ -2 & 0 & 5 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad \Sigma_3 = \begin{bmatrix} 6 & 4 & 5 \\ 4 & 6 & 3 \\ 5 & 3 & 9 \end{bmatrix}$$

g. Generate 500 samples for each of the three densities. Then augment each feature vector with 50 additional dimensions, each one from a Gaussian distribution $N(\mu = 1, \sigma = 5)$. This is because we would like to add noise to the data.
h. Compute the first principal components of the data, and generate a scatter plot of all projected samples.
i. Compute the Fisher's LDA projections of the data, and generate a scatter plot of all projected samples.
j. Discuss the previous results as well as your findings.

Note 1: Remember that σ is the standard deviation, not the variance.

Note 2: Please highlight samples by their labels.

Recommended MATLAB functions: `mvnrnd()`

6. K-NN for Digit Recognition: How Does It Tackle MNIST Challenge? **(12+6 Pts.)**

Keywords: *Classification Problem, Handwritten Digit Recognition, K-Nearest Neighbors, Distance Metrics*

Although **K-Nearest Neighbors** classification algorithm is not competitive in comparison to many supervised learning models, it is simple to explain and understand, and is usually praised for not having any assumptions about data. More than that, one may find it useful even in many real-world problems.

In this problem, you are to evaluate **K-NN** classification performance on the well-known USPS dataset, which contains a large number of handwritten digits in the images of the size 16×16. This dataset includes 1100 images for each digit. You can find this dataset in "inputs/P6" folder. Take 70% of the digit samples randomly as your training set, and the rest of them as your test set.

Figure 6 A few sample images of USPS dataset

a. Considering Euclidean metric, find 5 nearest neighbors for the first 3 test samples, and display them together.
b. Now considering the following metrics, find 5 nearest neighbors for the fourth test sample, plot them together and compare the results:
   b1. Euclidean distance
   b2. Manhattan distance
   b3. Chebyshev distance
   b4. Mahalanobis distance
   b5. Cosine distance
   b6. Hamming distance
   Hint: [1]
c. Considering Euclidean distance metric, find K-NN classifier precision on test digits. Set $k = 1,...,10$, and plot the precisions as a function of k. How does increasing k affects the precision?
d. Repeat part b. for all test images, and report the precision for each of the metrics. Comment on the results.

Note: Using built-in functions for K-NN classification (like `knnclassify()` or `fitcknn()` in MATLAB) or external libraries is not allowed in this problem. Please use your own implementation.

Recommended MATLAB functions: `reshape(), subplot()`

7. PCA and LDA in the Face Recognition Problem: Eigenfaces vs. Fisherfaces   (20+10 Pts.)

Keywords: *Classification Problem, Face Recognition, Principal Component Analysis (PCA), Linear Discriminant Functions (LDA), K-Nearest Neighbor Classifier, Fisherfaces, Eigenfaces*

Matthew Turk and *Alex Pentland* were the first to apply **Principal Component Analysis** (**PCS**) in **Face Recognition** problem. Their approach – introduced in 1991 and known as **Eigenfaces** – is considered as the first successful face recognition algorithm.

Fisherfaces, on the other hand, was an enhancement of the Eigenfaces method, which were proposed by *Joao Hespanha* in 1997. As can be guessed, it applies **Fisher's Linear Discriminant Analysis** (**FLDA** or **LDA**) for the **Dimensionality Reduction**.

(a)

(b)

Figure 7 Examples of Eigenfaces and Fisherfaces obtained in a certain dataset (a) First 7 Eigenfaces (b) First 7 Fisherfaces

In this problem, the goal is to implement Eigenfaces and Fisherfaces for recognising faces. You will be using AT&T dataset, which contains 400 images of the size 112×92. There are 40 distinct subjects, 10 images per each person. Here, 360 images are selected as the training set (9 per each subject) and 40 as the test set (one per each subject). Training set images and their corresponding labels are stored in 'trainset_imgs.csv' and 'trainset_labels.csv' respectively, while test set images and their labels are in 'testset_imgs.csv' and 'testset_labels.csv' respectively, all attached to 'inputs/P7' folder of this homework.

Figure 8 AT&T face images for a specific subject. As can be seen, the main variation in the face images is in their pose

First, let's see how Eigenfaces algorithm deals with this dataset.

a. Write a function `[W,mu]=eigenfaces_train(trainset,v)` which takes $n \times d$ train set images matrix `trainset`, where $n = 360$ is the number of training images, and $d = 10304$ is the number of pixels in each image. This function performs Principal Component Analysis on the data and computes the top `v` eigenvectors. It return the $v \times d$ matrix of eigenvectors `W`, and a d-dimensional vector `mu` representing the mean of the training faces. Compute the top 50 eigenvectors.

b. Reshape each of the top 50 eigenvectors obtained in the previous part, and display them by appending them together into a 1120×460 image (a grid of images containing 5 rows and 10 columns).

c. Select 10 random images from the train set, and show what each of these images would look like when using only the top v eigenvectors to reconstruct them, where $v = 1,...,10$.

    Hint: This reconstruction procedure should project each image into a v-dimensional space, project that v-dimensional space back into a 10304 dimensional space, and finally reshape that 10304 vector into a 112×92 image.

d. Now let's test the Eigenfaces method performance. Implement a function with the header `testset_labels=eigenfaces_test(trainset,trainlabels,testset,W,mu,v)` which takes the train set matrix `trainset` as well as their labels vector `trainlabels`, test set matrix `testset` and outputs of PCA, `W` and `mu`, and the number of eigenvectors to use `v`. Your function must first project each image from `trainset` and `testset` onto the space spanned by the first `v` eigenvectors. Then it must classify each test image using the nearest neighbor (1-NN) classifier by considering L₂ distance in the lower dimensional space. It must return an m-dimensional vector `testset_labels` encoding the predicted class label for each test face, where $m = 40$ is the number of test images. Evaluate `eigenfaces_test` on the test set images for values $v = 1,...,50$, and plot the error rates as a function of `v`.

e. Repeat the previous part, but throw away the first 5 eigenvectors. In other words, use `v` eigenvectors starting with the 6th eigenvector. Produce a plot similar to the one in the previous step, and comment on the result. How do you explain the difference in recognition performance in comparison with the previous part?

Next, it's time to examine Fisherfaces.

⭐ f. Implement a function `[W,mu]=fisherfaces_train(trainset,trainlabels,c)` that takes the same $n \times d$ train images matrix `trainset` as well as their corresponding labels vector `trainlabels`, and the number of classes $c = 40$. Your function must do the following:

-   Compute the mean `mu` of the training data, and use PCA to compute the first $n-c$ principal components. Let this be W_{PCA}.

-   Use the obtained W_{PCA} to project the training images into a $n-c$ dimensional space.

-   Compute the between-class scatter matrix S_B and the within-class scatter matrix S_W on the $n-c$ dimensional space from the previous space.

-   Compute W_{FLD} by solving for the generalised eigenvectors of the $c-1$ largest generalised eigenvalues for the problem $S_B w_i = \lambda_i S_W w_i$.

    Hint: In MATLAB, you can use the function `eig()` to solve for the generalised eigenvalues of S_B and S_W.

- The fisher bases will be a $W = W_{FLD}W_{PCA}$, where W is $(c-1) \times d$ dimensional, W_{FLD} is $(c-1) \times (n-c)$ dimensional, and W_{PCA} is $(n-c) \times d$ dimensional.

⭐ g. As in the Eigenfaces method, reshape the top 50 Fisher bases you obtained in the previous part into images of the size 112×92 and stack them into one big 1120×460 image.

⭐ h. Perform recognition on the test set with Fisherfaces. As in the Eigenfaces exercise, use a L₂ nearest neighbor classifier (1-NN), and evaluate results for each test images for values $v = 1, ..., 50$. Plot the error rates as a function of v.

Hint: Images of the given dataset are stored in `double` format. It means that if you want to use MATLAB function `imshow()` to display them, first you need to convert them to 8-bit unsigned integers, i.e. `uint8`.

Useful Sources: [1], [2], [3]

Recommended MATLAB functions: `csvread(), colormap(), imagesc(), reshape(), eigs(), svds(), eig(), subplot()`

---

8. Some Explanatory Questions   (8 Pts.) ✍

Please answer the following questions as clear as possible:

a. K-NN is said to be a *lazy learner*. How can you justify that?

b. What is the training error of K-NN, when $k = 1$, $k = 3$ and $k = \#\text{train samples}$.

c. One drawback in K-NN is that all of the k-nearest neighbors are weighted equally when deciding the class label, and it might be inappropriate for large values of k. Suggest a modification to the K-NN which can avoid it.

d. What would be a reasonable approach to determine the value of k in K-NN?

e. In PCA method, what would happen if the number of dimensions in the new space be equal to the number of dimensions in the original space?

f. When does PCA and LDA return the same projections? Explain.

Good Luck!
Ali Abbasi, Behnam Roshanfekr