Amirkabir University of Technology
(Tehran Polytechnic)

Assignment 3
Overview of Dimensionality Reduction and Supervised Learning

Homeworks Guidelines and Policies

- ***What you must hand in.*** It is expected that the students submit an assignment report (HW3_[student_id].pdf) as well as required source codes (.m or .py) into an archive file (HW3_[student_id].zip).
- ***Pay attention to problem types.*** Some problems are required to be solved *by hand* (shown by the icon), and some need to be implemented (shown by the icon). Please do not use implementation tools when it is asked to solve the problem by hand, otherwise you will be penalized and lose some points.
- ***Don't bother typing!*** You are free to solve by-hand problems on a paper and include their pictures in your report. Here, cleanness and readability are of high importance. Images should also have appropriate quality.
- ***Reports are critical.*** Your work will be evaluated mostly by the quality of your report. Do not forget to explain your answers clearly, and provide enough discussions when needed.
- ***Appearance matters!*** In each homework, 5 points (out of a possible 100) belong to compactness, expressiveness, and neatness of your report and codes.
- ***Python is also allowable.*** By default, we assume you implement your codes in MATLAB. If you are using Python, you have to use the equivalent functions when it is asked to use specific MATLAB functions.
- ***Be neat and tidy!*** 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, which must be named 'p3b.m'.
- ***Use bonus points to improve your score.*** Problems with bonus points are marked by the icon. These problems usually include uncovered related topics, or those that are only mentioned briefly in the class.
- ***Moodle access is essential.*** Make sure you have access to Moodle, because that is where all assignments as well as course announcements are posted. Homework submissions are also made through Moodle.

- ***Assignment Deadline.*** Please submit your work **before the end of December 31th**.
- ***Delay policy.*** During the semester, students are given 7 free late days which they can use them in their own ways. Afterwards, there will be a 25% penalty for every late day, and no more than three late days will be accepted.
- ***Collaboration policy.*** We encourage students to work together, share their findings, and utilize all the resources available. However you are not allowed to share codes/answers or use works from the past semesters. Violators will receive a zero for that particular problem.
- ***Any questions?*** If there is any question, please do not hesitate to contact us through the following email addresses: **ali.the.special@gmail.com** and **ebp.mohsen@gmail.com**.

1. Evaluating K-NN in Simple Classification (and Maybe Regression!) Problems (14+5 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	–2.2	–1.4	–1.0	–0.5	0.1	0.4	1.1	1.4	2.0	2.8
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.

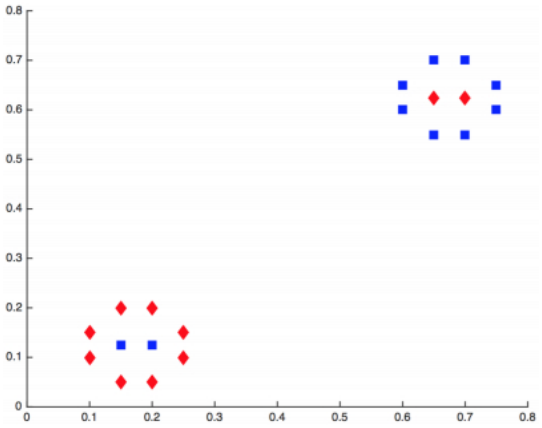

Figure 1 Data samples of part f. and g. with class 'blue' and class 'red'

In a different dataset, consider a binary classification task illustrated in Figure 2.

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.

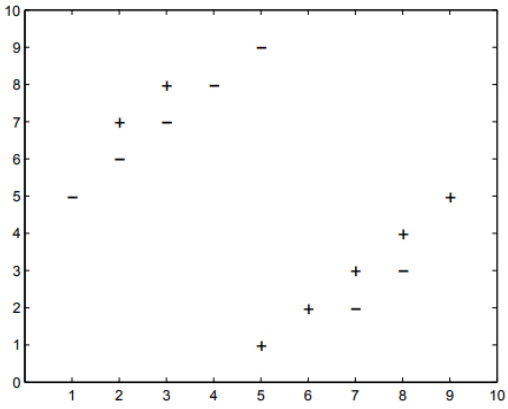

Figure 2 Dataset given for part h. to part l., with two classes '+' and '-'

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".

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 3. 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 $k=3$.

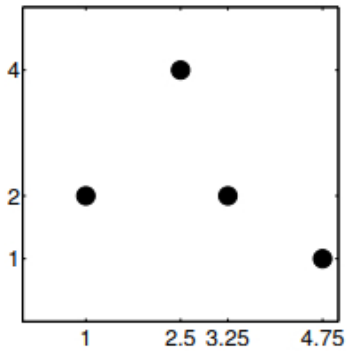

Figure 3 Data points given for K-NN regression problem

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

2. Studying the Role of Smoothing Parameter in Kernel Density Estimation (12 Pts.)

Keywords: *Density Estimation, Non-Parametric Methods, Kernel Density Estimation, Smoothing Parameter (Bandwidth)*

In **Kernel Density Estimation**, the **Smoothing Parameter** or **Bandwidth** controls the number of samples or window of samples used to estimate the probability for a new point. While a large window may result in a coarse density with little details, a small window may also have too much detail and not be smooth or general enough to correctly cover new or unseen examples.

In this problem, we are to investigate the importance of this parameter in practice. Consider the following two probability density functions:

$$p_1(x)=\begin{cases} x & \text{if } 0<x<1 \\ 2-x & \text{if } 1<x<2, \\ 0 & \text{otherwise} \end{cases} \qquad p_2(x)=xe^{-x^2/2}$$

a. Sketch the PDFs.

b. Generate N i.i.d samples from the given PDFs, assuming $N=\{10,100,1000\}$.

c. For a univariate Gaussian kernel, it is often recommended to select $h^* \approx 1.06\hat{\sigma}N^{-1/5}$, where h^* is the optimal choice of bandwidth, N is the number of samples and $\hat{\sigma}$ is the estimate of the standard deviation of the samples. Calculate the sample standard deviation, $\hat{\sigma}$. For each N, estimate the optimal value for bandwidth, $h^*(N)$.

d. Use kernel density estimation with a Gaussian kernel for each N to estimate the PDFs, considering three different bandwidth values $\{h^*(N)/3,\ h^*(N),\ 3*h^*(N)\}$.

e. Summarise your results by plotting the two PDFs estimates. For each of the given densities, you need to have 9 plots overall (18 in total). Overlay each plot with the groundtruth densities. Comment on the effects of h, N, and the kernel itself on the estimations you obtained.

f. **Recommended MATLAB Functions**: `rand()`, `subplot()`

3. K-NN In Action (I): Regression Analysis (12 Pts.)

Keywords: *Regression Problem, K-Nearest Neighbors*

K-Nearest Neighbors algorithm is probably the simplest widely used model in machine learning. In spite of its simplicity, **K-NN** has proven to be incredibly powerful in various machine learning applications. Although it's far more popular in classification problems, K-NN can also be used in any regression task. The aim of this problem is to investigate how K-NN can be equally effective when the target variable is continuous in nature.

First, consider a simple regression problem with one dependent variable and one independent variable. Here, the goal is to predict an animal's body weight given its brain weight. You will use "animals_weight.txt" dataset which contains a list of brain weight and body weight measurements from a bunch of animals.

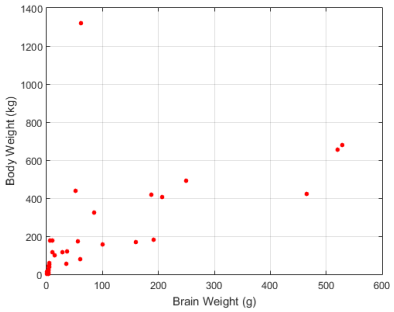

Figure 4 Distribution of brain and body weights of 64 species of animals, with two of the outlier samples removed for better visual purpose

a. Given the following brain weights of some unknown animals, predict their body weight using 1-NN, 3-NN and 5-NN models.

#Test Sample	Brain Weight
1	53.298
2	1247.122
3	0.583
4	4.859
5	0.041

b. Plot the regression line considering 1-NN, 3-NN and 5-NN models.

Now let's consider a more complicated problem, i.e. multivariate regression. You are given a dataset, "wine_quality.csv", containing 1600 red Vinho Verde wine samples. The goal is to model wine quality based on 11 physicochemical measurements.

Figure 5 The quality of red wine can somehow be evaluated by its physicochemical metrics such as the amount of citric acid or pH

c. Given the following table containing five set of physicochemical measurements, predict the corresponding wine qualities using 1-NN, 3-NN and 5-NN models.

#	Fixed acidity	Volatile acidity	Citric acid	Residual sugar	chlorides	Free sulfur dioxide	Total sulfur dioxide	density	pH	suphates	alcohol	quality
1	7.5	0.9	0.26	2.3	0.054	19	13	0.99708	3.78	0.55	9.7	?
2	5.4	0.78	0.17	3.2	0.084	11	58	0.9987	2.94	0.83	11.8	?
3	8.2	0.56	0.46	1.7	0.069	25	15	0.997	3.39	0.65	12.5	?
4	6.0	0.7	0.01	4.6	0.093	6	104	0.99746	3.12	0.52	10.5	?
5	10.8	0.43	0.31	2.5	0.105	35	31	1.0001	3.22	0.48	11.1	?

4. K-NN In Action (II): Image Segmentation (14 Pts.)

Keywords: *Classification Problem, Image Segmentation, K-Nearest Neighbors*

Following previous problem, here we are going to evaluate **K-NN** classification capabilities in a different area. Imagine an **Image Segmentation** task where the goal is to detect and separate road regions of an input image. Such a system is necessary in many traffic-related applications, including lane departure warning system.

Here, two frames of a rural highway are available, Figure 6 part a and b. You are given a ground truth segmented version of frame a, which can somehow be used in the training phase of a K-NN model.

a. Use 1-NN, 5-NN and 9-NN models to apply image segmentation on the image of the frame b. Display and compare the obtained results.
b. Do you think K-NN can be used in a lane detection system? Explain based on the observations you made in this problem.

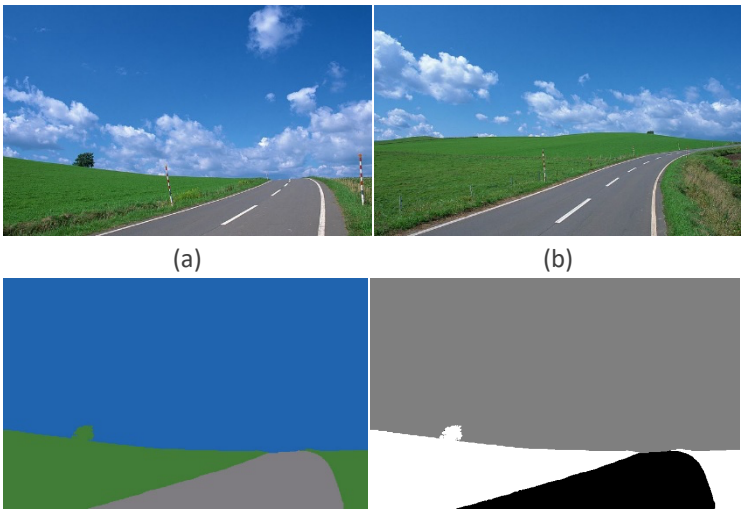

Figure 6 Two frames of a rural highway alongside the segmented version and segmentation mask of the first (a) frame a, which is used in training phase (b) frame b, which is used in test phase (c) segmented version of frame a (d) segmentation mask corresponding to frame a

5. Wrestling with Support Vector Machines in MATLAB (15 Pts.)

Keywords: *Classification Problem, Supervised Learning, Support Vector Machine, Kernel Trick*

Support Vector Machine (**SVM**) is a linear model which can solve linear and non-linear problems with a simple idea; it creates a line or a hyperplane which separates the data into classes. There are many possible choices for this line or hyperplane, and SVM attempts to find a plane that has the maximum **Margin**, which is the distance between data points of both classes. Data points that are closer to this hyperplane and influence on the position and orientation of the hyperplane are called **Support Vectors**.

The goal of this problem is to study SVM in practice. But don't worry, you don't have to write any code. Instead, a straightforward implementation of a support vector machine is given to you and you are expected to complete its missing parts properly. Note that the code is written in MATLAB.

a. The function `classify_grid()` applies a learned SVM classifier which is needed to test a classifier. Fill the line 41 of this function with an appropriate statement.
b. Fill the line 47 of the function `trainsvm()`.
c. Now fill the line 77 of the function `trainsvm()`. Solve the SVM optimisation problem by using the quadprog solver in MATLAB.
d. Load the file "test1.dat" and run the following command:
```
[SV,alpha,b] = trainsvm(D,inf,@kernel_linear,[]);
```
You should end up seeing figures like those in Figure 7. Explain the parameters that the function returns.

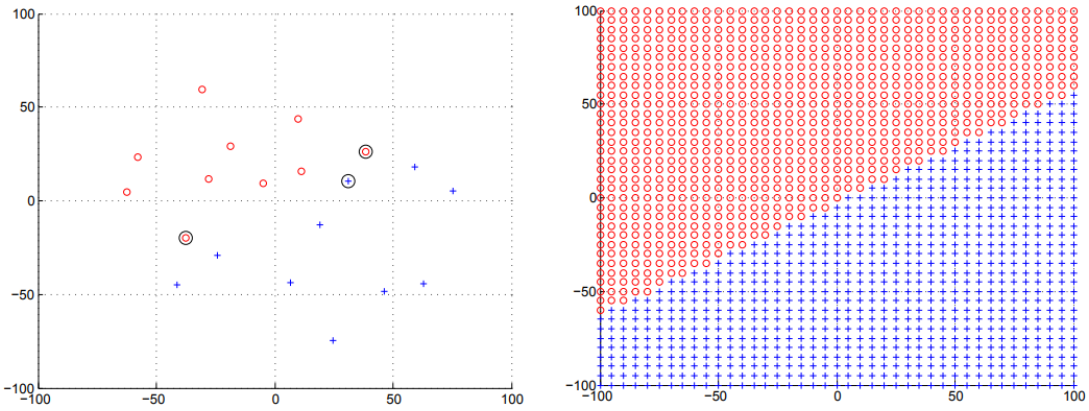

Figure 7 You are expected to obtain these two figures after filling the missing parts in the given code

e. What are the circled points in Figure 7? Explain.
f. What is the `inf` parameter in the inputs of the function `trainsvm()`?
g. Implement a modified kernel function that performs a quadratic kernel. Load the file "nonlin1.dat" and run the SVM. Plot and explain the output.
h. Now load the file "noise1.dat" and run the code. Set the second parameter to `inf`, 10, 1 and 0.1 and plot the obtained results. What's your observations? Specify the role of this parameter.

6. PCA or slope detector? (18 Pts.)

Keywords: *Principal Component Analysis (PCA), Dimension Reduction, Orientation Detection, Image Binarize*

One of the most critical tasks in **Image Preprocessing** is extracting the slope of objects according to the horizontal or vertical axis. You will be faced with implementing a PCA-based solution to this problem.

a. Load RGB *base.jpg* image and convert it to a grayscale single channel image. Different approaches exist that address the mentioned conversation. One of them is averaging the triple channels.
b. Now, you should detect pixels of the watch. To aim this, with a threshold, convert the grayscale image to a binary image(0 or 255).
c. You map the watch`s pixels to $R^2$ space in this step. You can extract horizontal and vertical indexes of pixels that are in the watch`s area. Now you have a dataset that has two features (indexes). plot the obtained dataset.
d. Implement the PCA and perform it on the dataset and obtain eigenvectors and eigenvalues; plot the obtained eigenvectors and points in one graph. Discuss the eigenvalues and corresponding eigenvectors.
e. Using the eigenvectors, calculate the slope of the watch based on the horizontal axis. repeat these steps for all test images.
f. Is this strategy robust against of detect slopes in the same direction? why? try to suggest a solution and implement it.

7. Some Explanatory Questions (10 Pts.)

Please answer the following questions as clear as possible:

a. In Parzen window density estimation, is there a method to find an appropriate value for bandwidth? Explain.
b. Is k-NN decision boundary always linear? Justify your answers with examples.
c. What is the relationship between the values of k and the smoothness of the decision boundary of a k-NN classifier?
d. Consider a 2-class classification problem in which there are samples of both classes in the training set. Is it possible for a 1-NN classifier to always assign a specific label to all test samples? If yes, give an example. If no, explain.
e. Under what circumstances do you recommend using SVM instead of MLP? What about the reverse? Which problems do both SVM and MLP suffer to solve?

Good Luck!

Ali Abbasi, Mohsen Ebadpour