# Assignment 5
## Supervised Learning vs. Unsupervised Learning: A Deeper Look

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: HW5_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 January 12th**.
10. If there is ***any*** question, please don't hesitate to contact me through the following email address:
    - **ali.the.special@gmail.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. Linear Discriminant Analysis, Fisher Criterion, MSE and Beyond (14 Pts.)

**Keywords**: *Classification Problem, Supervised Learning, Linear Discriminant Analysis (LDA), Fisher Criterion, Minimum-Squared Error (MSE)*

**Linear Discriminant Analysis** (**LDA**) is a method in machine learning which tries to find a linear combination of features that separate two or more classes of objects. A **Linear Discriminant Function** divides the feature space by a hyperplane decision surface, where the orientation of such a surface is determined by the **Weights** vector (i.e. normal vector $w$), and the location of the surface is determined by the **Bias** ( $w_0$ ).

In this problem, the goal is to get you more familiar with **LDA** as well as the **Fisher Criterion**. You will also examine more linear discriminant functions in the following problems.

Assume a two-category classification problem, where there are two sets of data with normal distribution such that:

$$P_1 = P_2 = 0.5, \qquad \mu_1 = \begin{bmatrix} -3 \\ 0 \end{bmatrix}, \mu_2 = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \qquad \Sigma_1 = \begin{bmatrix} 1.5 & 1 \\ 1 & 1.5 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 1.5 & -1 \\ -1 & 1.5 \end{bmatrix}$$

a. Find the linear discriminant function which maximises the Fisher criterion.
b. Minimise the error by adjusting the threshold.
c. Use the linear discriminant function to classify the point $x = [-0.5, 0.5]^T$.
d. Generate 500 samples for each class (1000 in total) and sketch the separating line obtained in part a.

Now in a different scenario, assume data samples are given in the following table. These samples are from two classes with equal prior probability.

| $x_1$ | $x_2$ | Class |
|---|---|---|
| $-1$ | $-1$ | 2 |
| $-1$ | 0 | 2 |
| $-1$ | 1 | 2 |
| 0 | $-1$ | 2 |
| 0 | 1 | 2 |
| 1 | $-1$ | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 2 |

e. Design a linear classifier which minimises the mean-square error.
f. Sketch the separating line resulted in part d.

**Recommended MATLAB functions**: `plot()`, `fplot()`

## 2. Representing Logical Functions Using Perceptron Algorithm (12+4 Pts.)

**Keywords**: *Classification Problem, Supervised Learning, Neural Networks, Perceptron Rule*

The **Perceptron Algorithm**, invented by *Frank Rosenblatt* in 1957, is a linear classifier that can be used for supervised learning of binary classifications. Hence, it may be used to perform various logic functions.

Here, we will explore the expressiveness of neural networks, by examining their ability to represent Boolean functions. Assume the inputs $X_i$ will be $0$ or $1$, and the activation function be a sigmoid function. The output $Y$ will be real-valued, and interpreted as a Boolean value $1$ if $Y > 0.5$, and $0$ otherwise.

a. Demonstrate the implementation of the logical function COMPLEMENT ($Y = \neg X_1$) using a single perceptron unit.

b. Determine 3 weights for a single perceptron unit with two inputs $X_1$ and $X_2$, that implements the logical OR function ($Y = X_1 \vee X_2$).

c. Is it possible to do the same for the logical AND function ($Y = X_1 \wedge X_2$) in a single perceptron unit? If so, determine the weights, and if not, explain why.

d. It is not possible to represent the EXCLUSIVE-OR function ($Y = X_1 \oplus X_2$) in a single perceptron unit. However, it would be possible to do so by applying multiple perceptron units. Implement this function by using the smallest number of units possible. Draw your network and display all weights.

e. Design a neural network with only one hidden layer (with any number of units) that represents $(B \Rightarrow A) \oplus (C \Rightarrow \neg D)$. Draw your network and display all weights.

## 3. Widrow-Hoff Weights Update Rule in a Regression Problem (8 Pts.)

**Keywords**: *Regression Problem, Supervised Learning, Neural Networks, Weights Update, Widrow-Hoff Algorithm, Least-Mean-Squared Rule, Stochastic Gradient Decent*

**Widrow-Hoff Algorithm** or **Least-Mean-Squared (LMS) Rule** is a **Stochastic Gradient Descent** method used for updating **Neural Networks** weights in that the weights are only updated based on the error at the current time. It was introduced by *Bernard Widrow* and his Ph.D. student *Ted Hoff*, back in 1960.

In this problem, we are going to take a closer look at this rule by designing a neural network which is able to represent a simple regression problem. Consider the following dataset:

$$X = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 2 \\ 1 & 2 \\ 2 & 1 \end{bmatrix}, y = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix}$$

a. Find the linear discriminant function by applying Widrow-Hoff algorithm (batch version) which minimizes

$$L_r(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i + b - y_i)^2$$

Consider the following initialisation: $\mathbf{w} = [0.1, 0.1]^T$, $b = 0.1$, $\eta = 0.1$, $\theta = 0.06$

b. Plot the data points and the decision boundary at each iteration.

**Recommended MATLAB functions**: `pause()`

## 4. Getting Familiar with Support Vector Machines (12 Pts.)

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

**Support Vector Machines** (**SVM**) is a known discriminative model with associated learning algorithms which formally defined by a separating hyperplane. Introduced by *Vladimir N. Vapnik* in 1963, SVM algorithm tries to find an optimal hyperplane in a high – or infinite – dimensional space, which can be used for classification, regression or other tasks like outlier detection. It is inspired by the idea that a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class (called **Functional Margin**).

Trying to avoid Support Vector Machines technicalities, here we are going to explore SVM algorithm on two toy datasets.

First, assume a simple 1-D classification problem, with a dataset $X$ of samples which defined by a single input feature, and a desired classification output $Y$ such that $Y \in \{-1, +1\}$. The dataset $X$ contains three samples $x_1 = -2$, $x_2 = 2$ and $x_3 = 3$ in class $-1$, and two samples $x_4 = -1$ and $x_5 = 1$ in class $+1$.

a. Plot the samples in 1-D space. Are they linearly separable?
b. Map the features into 2-D space using the following mapping function: $\phi(X) = (X, X^2)$ Plot the transformed data as well as the support vectors.
c. Manually plot the maximum-margin separating line by visual inspection of the transformed data points. This would be a line in 2-D space which can be parameterised by the equation $w_1 p + w_2 q + c = 0$, where $(p, q)$ is a transformed point. Find $w_1$, $w_2$ and $c$.
d. Find the margin for the classifier.

Now, Let's face a similar problem in 2-D space, where the goal is to classify the given dataset using linearly separable SVM:

| # | $X_1$ | $X_2$ | $y$ |
|---|---|---|---|
| *1* | 3 | 4 | *Red* |
| *2* | 2 | 2 | *Red* |
| *3* | 4 | 4 | *Red* |
| *4* | 1 | 4 | *Red* |
| *5* | 2 | 1 | *Blue* |
| *6* | 4 | 3 | *Blue* |
| *7* | 4 | 1 | *Blue* |

e. Sketch the samples, as well as the optimal separating hyperplane and provide the equation of the form $b + \omega_1 X_1 = \omega_2 X_2 = 0$ for this hyperplane.
f. Describe the decision rule for maximal margin classifier. It must be something like "Classify to Blue if $b + \omega_1 X_1 = \omega_2 X_2 \geq 0$ and classify to Red otherwise". Find $\mathbf{w}$ and $b$.
g. Indicate the margin for the hyperplane on your sketch from the part e.
h. Indicate the support vectors for the classifier.
i. Argue that whether a slight movement of the seventh sample would affect the hyperplane of the linear SVM or not.
j. Sketch a non-optimal hyperplane which separates the data, and provide an equation for it.
k. Draw a new observation on the plot, so that the samples of the two classes are no more separable by a hyperplane. Label that point clearly.

## 5. Clustering; the Idea of Classifying Data in an Unsupervised Fashion (12 Pts.)

**Keywords**: *Clustering Problem, Unsupervised Learning, K-Means Clustering, Hierarchical Clustering, Dendrogram*

**Clustering** algorithms are probably the most common examples of **Unsupervised Learning** methods, in which the goal is to group a set of objects in such a way that objects in the same group, i.e. **Cluster**, are more similar according to a certain metric.

One of the most popular methods of clustering is **K-Means**, where the clustering process of an observation is done based on the nearest clusters mean. **Hierarchical Clustering**, on the other hand, is a different method of clustering which tries to build a hierarchy of clusters, usually presented in a **Dendrogram**.

In this problem, we are going to walk through the clustering process by solving some examples by hand. First, assume the dataset $X$ as follows:

$$X = \begin{bmatrix} 4.6 & 2.9 \\ 4.7 & 3.2 \\ 4.9 & 3.1 \\ 5.0 & 3.0 \\ 5.1 & 3.8 \\ 5.5 & 4.2 \\ 5.9 & 3.2 \\ 6.0 & 3.0 \\ 6.2 & 2.8 \\ 6.7 & 3.1 \end{bmatrix}$$

You are asked to perform a K-Means clustering on the given dataset, where $k$ is set to 3 and the centres of 3 clusters are initialised as $\mu_1 = (6.2, 3.2)$ (red), $\mu_2 = (6.6, 3.7)$ (green) and $\mu_3 = (6.5, 3.0)$ (blue). Use the Euclidean distance as the distance function.

a. What would be the centre of the cluster 1 (red) after the first iteration?
b. What would be the centre of the cluster 2 (green) after the second iteration?
c. What would be the centre of the cluster 3 (blue) when the algorithm converges?
d. How many iterations are required for the algorithm to converge?

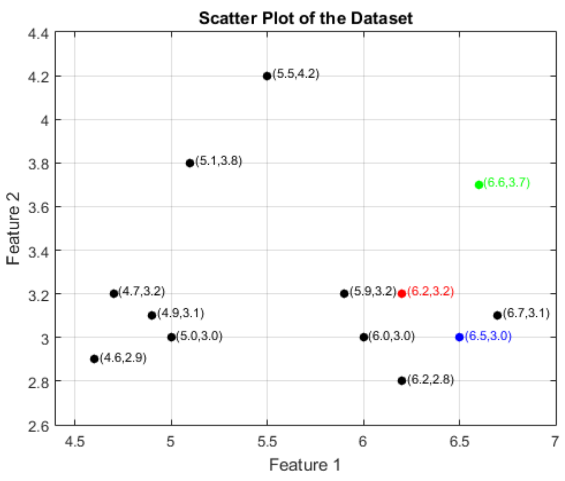

*Figure 1 Scatter plot of the dataset points (black) as well as the initial centre points (red, green and blue)*

Now, assume you are applying K-Means with $k = 2$ clusters on the following set of 1-D points:

$$-2, 0, 10$$

e. Considering the fact that K-Means can get stuck in local optima, describe the precise conditions on the initialisation of $\mu_1 \in \mathbb{R}$ and $\mu_2 \in \mathbb{R}$ such that running K-Means will yield the global optimum of the objective function. Assume $\mu_1 \le \mu_2$, and if in the first step of K-Means, no points are assigned to a cluster $j$, then in the next step $\mu_j$ is set to $\infty$.

**Hint**: To get some intuition, try running K-Means with several different initialisations.

Finally, assume the given similarity matrix below:

| | **P₁** | **P₂** | **P₃** | **P₄** | **P₅** | **P₆** |
|---|---|---|---|---|---|---|
| **$P_1$** | 1.0 | 0.55 | 0.44 | 0.57 | 0.35 | 0.47 |
| **$P_2$** | 0.55 | 1.0 | 0.42 | 0.53 | 0.47 | 0.46 |
| **$P_3$** | 0.44 | 0.42 | 1.0 | 0.40 | 0.53 | 0.52 |
| **$P_4$** | 0.57 | 0.53 | 0.40 | 1.0 | 0.50 | 0.51 |
| **$P_5$** | 0.35 | 0.47 | 0.53 | 0.50 | 1.0 | 0.49 |
| **$P_6$** | 0.47 | 0.46 | 0.52 | 0.51 | 0.49 | 1.0 |

f. Perform a single linkage hierarchical clustering.

g. Perform a complete linkage hierarchical clustering.

h. Perform an average linkage hierarchical clustering.

## 6. More Into Clustering: K-Medoids, Elbow Methods and So on (14+6 Pts.)

**Keywords**: *Clustering Problem, Unsupervised Learning, K-Means Clustering, K-Medoids Clustering, Hierarchical Clustering, Dendrogram, Elbow Method, Purity Measure, Normalised Mutual Information (NMI)*

Following the previous problem, here you are going to investigate **Clustering** more in detail. You are also going to get familiar with several new clustering concepts. All datasets can be found in “inputs/P6” folder attached to this file.

In the first part, you are working with a random generated two-dimensional dataset, “rand_ds.csv”, shown in Figure 2.

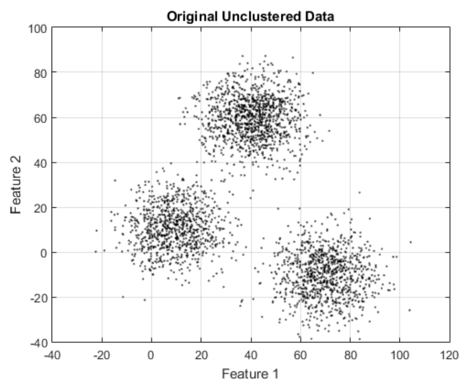

*Figure 2 Sample points of the given random dataset*

a. Perform clustering on the given data by applying K-Means method with parameter $k$ set to 2, 3, 5 and 7. In each case, plot the clustered data and highlight each cluster with a specific color. Also, indicate the final centres with a unique shape on your plot.

b. In more complicated datasets, finding a proper value for $k$ is often challenging. There are various methods one can use to determine a good value for $k$, such as **Elbow Method**, which is probably the most known method in this area. Run the algorithm for values of $k = 10, \dots, 1$, and plot the $k$ values against the **Sum of Squared Errors** (**SSE**). How can this plot help you finding a good value for $k$? What would be that value?

c. **K-Medoids** is one of the many variations of K-Means, which is fundamentally similar to it, with only slight differences. Repeat part a. for the case that K-Medoids clustering is applied.

Next, read the dataset “gene_ex.csv“, which contains the data on 40 tissue samples with measurements on 1000 genes. The first 20 samples are from healthy patients, while the second 20 are from diseased patients.

d. Perform hierarchical clustering on the samples, once using Euclidean and once using correlation-based distance. Plot the dendrogram. Do the genes separate samples into two groups? Do your results depend on the type on linkage used?

e. We would like to know which genes differ the most across two groups. Try to answer this question.

Finally, you are to evaluate the clustering performance of K-Means versus hierarchical methods on the famous two-spiral dataset. It contains 700 samples, each with sample coordinate points and the corresponding cluster label, and can be found in the file “two_spiral.csv“.

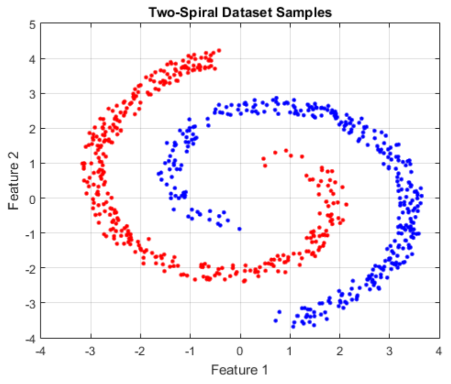

*Figure 3 Scatter plot of “two-spiral” samples*

f. Compare K-Means method with a hierarchical clustering method in terms of **Purity** and **Normalised Mutual Information (NMI)** measures. For hierarchical clustering, use a bottom up way and for cluster merging use both single linkage and complete linkage methods.
**Note**: You are free to use ready-to-use codes (like this) for clustering evaluation measures.

**Note**: Please use your own implementation of clustering algorithms.

**Recommended MATLAB functions**: `csvread(), dist(), cor(), dendrogram()`

## 7. Doing Some Image Processing Using K-Means (15 Pts.)

**Keywords**: *Clustering Problem, Supervised Learning, K-Means Clustering, Image Segmentation, Image Compression*

Although **K-Means** might sound like a basic **Clustering** method, it has various applications in different areas, from machine learning and computer vision to astronomy and even agriculture. As an example, one can count several image processing problems that can be easily dealt with using K-Means method.

In this problem, you are to examine K-Means on two image processing task, namely **Image Segmentation** and **Image Compression**. The first deals with algorithms and methods which try to partition a digital image into several parts in order to change it into a more meaningful and easier to analyse representation, while the second handles the process of reducing the cost for storage or transmission of digital images.

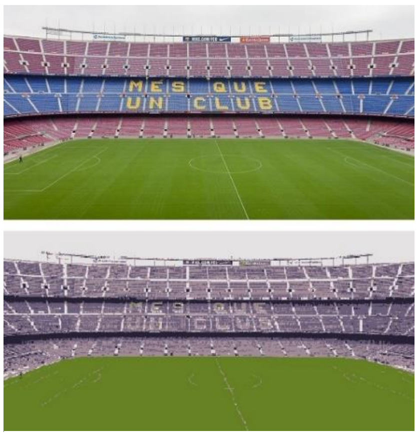

*Figure 4 An example of image segmentation, with initial image (Top) and the result of k-means segmentation with k set to 5 (Bottom)*

Let's start with a grayscale image segmentation. In this part, you are working with the image "trump_new_hairstyle.png" located in "inputs/P7" folder.

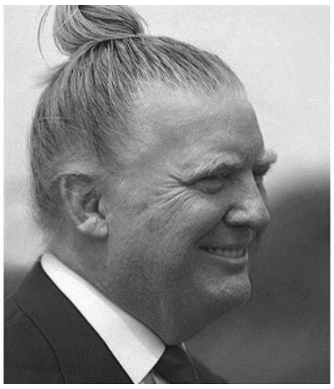

*Figure 5 The grayscale input image given for part a.*

a. Implement a function with a header as follows:
`[centres,mask]=kmeans_seg_gray(img_gray,k)`
This function takes a $n \times m$ grayscale image `img_gray` and a constant value `k`, and applies K-Means algorithm upon the input image in such a way that it returns a $nm \times 1$ vector `centres` which stores final centre points, and a $n \times m$ matrix `mask` which determines which of the `k` clusters each pixel belongs to. Now apply it on the given image, and using the outputs `centres` and `mask`, create a new $n \times m$ image in which the value of every pixel is set to its cluster centre. Display the results for $k = 3,5,7,9$. Attach the resultant images to your report as well.

**Hint**: Here, you must treat each pixel value as a point in 1-D space.

Now, let's deal with a color image in a similar fashion. Read the image "bald_donald.png", located in the same folder.

*Figure 6 An image given for color image segmentation*

b. Implement a RGB version of the function in part a. with the following header:

`[centres,mask]=kmeans_seg_rgb(img_rgb,k)`

It takes a $n \times m \times 3$ color image `img_rgb` and a constant value `k`, and works exactly the same as the previous function, only the output variable `centres` is now a matrix of the size $nm \times 3$. Once again, apply your function upon the given input image, and display the segmentation results for $k = 3,5,7,9$, and attach them to your report too.

Finally, read the image "leo.png" in the mentioned folder.

*Figure 7 A 600x900 input image, with the approximate size of 1Mb, given for part c.*

c. Apply K-Means image segmentation on the input image, with the variable $k$ set to $k = 32,64,128,192$. Display the segmented images. Can you notice any difference? Save the images (not figures) and report their sizes, and discuss whether K-Means can be useful in image compression or not.

**Note 1**: This part might sound a bit tedious, but it will worth the effort.

**Note2**: Save your results in the same format as the input image (.png) so that the comparison becomes meaningful.

**Note**: Please use your own implementation of K-Means algorithm.

**Recommended MATLAB functions**: `im2double()`, `uint8()`, `subplot()`, `imwrite()`, `reshape()`

## 8. Some Explanatory Questions (8 Pts.)

Please answer the following questions as clear as possible:

a. Does a neural network with a single hidden layer capable of separating non-convex regions? Why / Why not?
b. If samples from two classes are distinct (there is no feature point which is labelled by both classes), is there always a nonlinear mapping to a higher dimension which is able to leave the points linearly separable? Why / Why not?
c. Considering a linear soft-margin SVM, if no free support vector is found after training, can you conclude that the data is not linearly separable? Justify your answer.
d. What advantages and disadvantages do support vector machines have versus multilayer perceptrons? What problems both SVMs and MLPs suffer from?
e. If we run K-Means algorithm several times with the same number of clusters but different starting point, does it always converge to the same solution? Why / Why not?
f. Can K-Means with a specific parameter $k$ ever converge to a result which contains more or less than $k$ clusters? Explain.

*Good Luck!*
*Ali Abbasi*