# Statistical Pattern Recognition
## Assignment 6

Ali Gholami

Department of Computer Engineering & Information Technology
Amirkabir University of Technology

*https://aligholamee.github.io*
*aligholami7596@gmail.com*

**Abstract**

In this report, we'll analyze the different techniques of linear classification such as *Perceptron Learning Rule* and *Support Vector Machine*. More *in-depth* material about linear discriminant functions is already covered in the class. For the final section of this report, we'll delve into the *Clustering* techniques such as *Flat Clustering* and *Hierarchical Clustering* to solve problems arising in the *Unsupervised Learning* context.

**Keywords.** *LDF, Perceptron, Ho-Kashyap, Widrow-Hoff, Support Vector Machine, Flat Clustering, Hierarchical Clustering.*

# 1 LDF Based on Perceptron Learning Rule

Recall that a perceptron learns a linear classifier with weight vector $W$. It predicts:

$$\hat{y} = sign(W^T x_t)$$

assuming here that $\hat{y} \in \{+1, -1\}$. When the perceptron makes a mistake, it updates the weights using the formula:

$$W^{t+1} = W^t + y_t x_t$$

Imagine that we have $x_t \in I^2$, and we encounter the following data points:

| X[1] | X[2] | y |
|---|---|---|
| 1 | 1 | 1 |
| 2 | -1 | -1 |
| -3 | -1 | -1 |
| -3 | 1 | 1 |

(a) Starting with $W = [0\ 0]^T$, use the Perceptron Algorithm to learn on the data points in the order from top to bottom. Show the Perceptron's linear decision boundary after observing each data point. Be sure to show which side is classified as positive.

(b) Does our learned perceptron maximize the margin between the training data and the decision boundary? If not, draw the maximum-margin decision boundary.

## Solution

We'll start by augmenting the feature vectors given in the table. We'll add a new feature $x_0$ (bias term) and set it to 1 for class 1, and $-1$ for class $-1$. This results in the following data points:

| Y[0] | Y[1] | Y[2] | Class |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| -1 | -2 | 1 | -1 |
| -1 | 3 | 1 | -1 |
| 1 | -3 | 1 | 1 |

we'll then augment the weight vector for a consistent definition of the decision boundary. The yields in

$$W = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow a^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Note that according to the Perceptron Learning Rule, we're looking for a line to meet the following constraint:

$$z_i = a^T y_i > 0 \;\; \forall y_i \tag{1.1}$$

weights $a = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ cannot correctly classify the first point since $z_1^0 = 0$ which is a sign of misclassification. We can correctly classify this point conducting a *weight update* using Perceptron Learning Rule:

$$a^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

The resulting linear boundary will incorrectly classify the second point:

$$z_2^2 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} . \begin{bmatrix} -1 \\ -2 \\ 1 \end{bmatrix} \leq 0$$

Thus we have to update the weights using Perceptron Rule again:

$$a^2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}$$

The resulting line will poorly perform on the third data point:

$$z_3^3 = \begin{bmatrix} -1 & 3 & 1 \end{bmatrix} . \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix} \leq 0$$

Updated weights for the third iteration is provided below:

$$a^3 = \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix} + \begin{bmatrix} -1 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 3 \end{bmatrix}$$

following the same pattern for the final data point results in the final weight update:

$$a^4 = \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ -3 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ -4 \\ 3 \end{bmatrix}$$

Corresponding decision boundaries for each step is provided in the figure 1.1

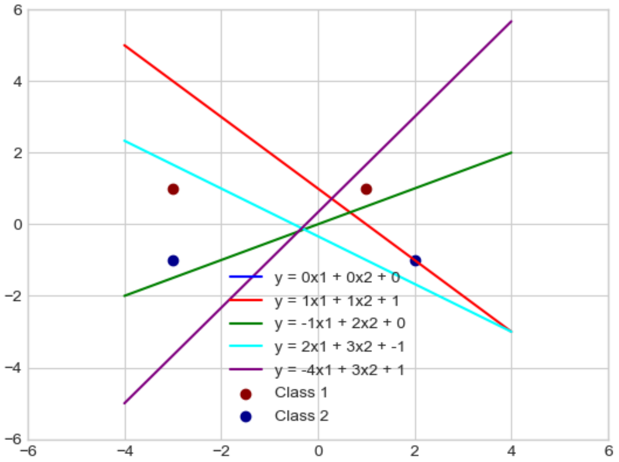


Figure 1.1: Illustration of Perceptron decision boundaries and class points.

The learned Perceptron does not maximize the margin between the training data and the decision boundary. The max-margin linear discriminator is $y = 0$.

## 2 Perceptron in Practice

Start with the truth tables for boolean *AND* and *OR*. Your goal is to write an small program to train a *two-input one-output* Perceptron using this data. Select some random initial weights in the interval $(0, 1)$. Use a learning parameter 0.1. The stopping condition is 50 iterations or *no change in weights from one epoch to the next*, whichever comes first.

(a) Encode *FALSE = 0* and *TRUE = 1* for the input values. Remember that the output of a Perceptron, by definition is $+1$ or $-1$.

(b) Encode *FALSE = -1* and *TRUE = +1* for the input values. Remember that the output of a Perceptron, by definition is $+1$ or $-1$.

For each case, please show the evolution in the change of weights by arranging them in a nice tabular format. Remember to include the weight $w_0$ as a part of the training process and set your $x_0$ to $+1$ all the time. For each case, please plot the separating hyperplane in the $x_1 - x_2$ plane. Submit your code, result of your run and your comments.

## Solution

(a) In this section, we have provided any code in the *src* folder. The code is written in Python and the results is provided here. As can be seen from figure 2.1, the separating line correctly evolves until all of the points are correctly on one side of the decision boundary only. Since the negative class is normalized before starting the algorithm, rolling back the changes on data points would illustrate how 2 classes are separated. In this simple case, points are linearly separable. Thus, Perceptron can find the separating line without difficulties. The problem arises when the classes are not linearly separable. This case is observed in the next figure where after 50 iterations, the line cannot separate the classes. Don't bother, it can't be done mathematically either!

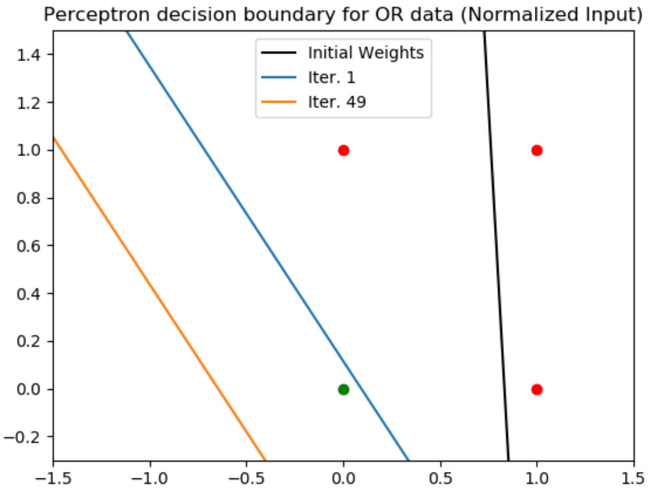


Figure 2.1: Illustration of Perceptron decision boundaries and *OR* data points.

### Caution!

All points are plotted in *normalized* mode; Points from class 2 are multiplied by a *-1*. As can be seen in figure 2.2, Perceptron cannot find a proper separating line for *XOR* data points. In case of *or* problem, we've also provided the tabular format of the weight updating procedure in figure 2.3.

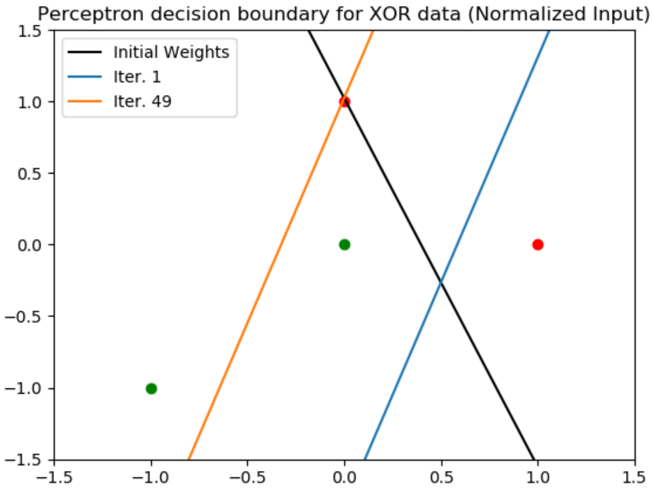
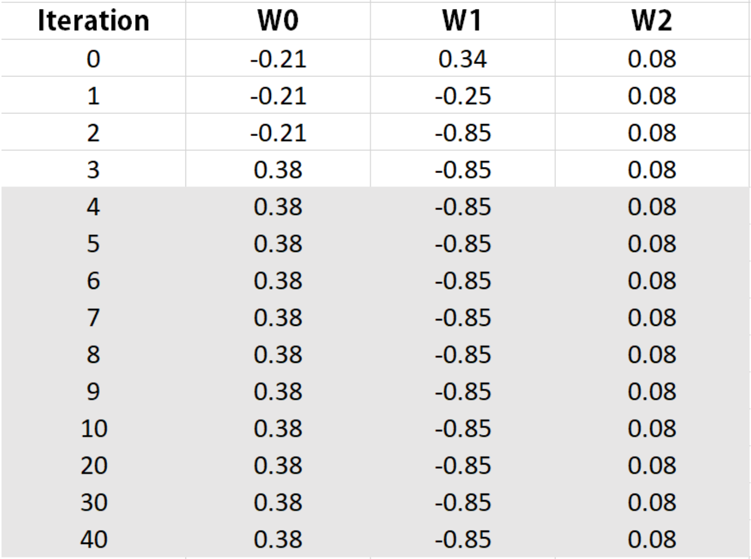


Figure 2.2: Illustration of Perceptron decision boundaries and *XOR* data points.



| Iteration | W0 | W1 | W2 |
|---|---|---|---|
| 0 | -0.21 | 0.34 | 0.08 |
| 1 | -0.21 | -0.25 | 0.08 |
| 2 | -0.21 | -0.85 | 0.08 |
| 3 | 0.38 | -0.85 | 0.08 |
| 4 | 0.38 | -0.85 | 0.08 |
| 5 | 0.38 | -0.85 | 0.08 |
| 6 | 0.38 | -0.85 | 0.08 |
| 7 | 0.38 | -0.85 | 0.08 |
| 8 | 0.38 | -0.85 | 0.08 |
| 9 | 0.38 | -0.85 | 0.08 |
| 10 | 0.38 | -0.85 | 0.08 |
| 20 | 0.38 | -0.85 | 0.08 |
| 30 | 0.38 | -0.85 | 0.08 |
| 40 | 0.38 | -0.85 | 0.08 |

Figure 2.3: Illustration of decision boundary weights update in OR problem.($\eta = 0.6$ for faster convergence)

(b) Figure 2.4 provides illustrative information on the resulting decision boundary. figure 2.5 provides the decision boundaries for the XOR problem in second case.

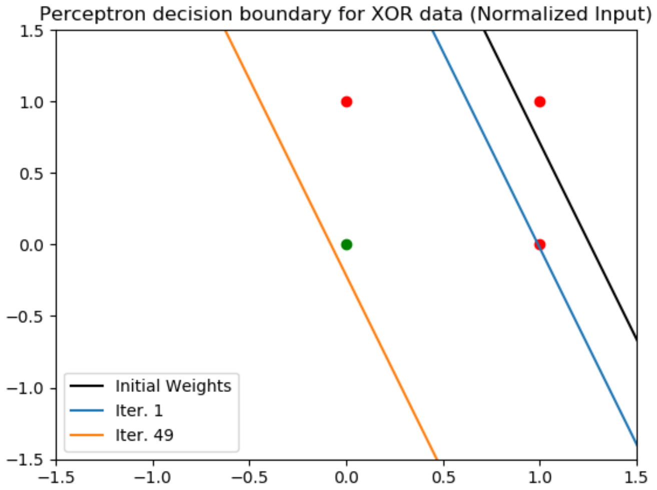


Figure 2.4: Illustration of decision boundary weights update in OR problem.(False = -1)

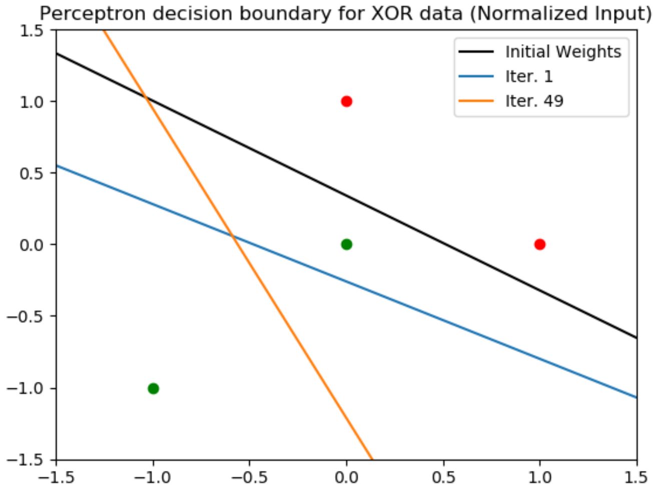


Figure 2.5: Illustration of decision boundary weights update in OR problem.(False = -1)

# 3 Support Vector Machine

Consider a dataset with 2 points in $1D$:

$$X_1 = 0,\ Y_1 = -1 \qquad X_2 = \sqrt{2},\ Y_2 = 1$$

Consider mapping each point to $3D$ using the feature vector $\phi(x) = [1,\ \sqrt{2}x,\ x^2]^T$. Here, the max margin classifier has the form:

$$\hat{W},\ \hat{W}_0 = argmin||W||^2 \tag{3.1}$$

so that

$$y_1(W^T\phi_1(x) + W_0) \geq 1$$
$$y_2(W^T\phi_2(x) + W_0) \geq 1$$

(a) Write down a vector that is parallel to the optimal vector $\hat{W}$. Hint: $\hat{W}$ is prepependicular to the decision boundary between the two points in the $3D$ feature space.

(b) What is the value of the margin that is achieved by this $\hat{W}$? Hint: The margin is the distance from each support vector to the decision boundary. Think about the geometry of the points in feature space, and the vector between them.

(c) Solve for $\hat{W}$, using the fact that the margin is equal to $\frac{1}{||\hat{W}||}$.

(d) Solve for $\hat{W_0}$ using your value for $\hat{W}$ and (3.1). Hint: The points will be on the decision boundary, so the inequalities will be tight. A *Tight Inequality* is an equality that is as strict as possible.

## Solution

The initial 1-dimensional dataset is $D = (0 \quad \sqrt{2})$. Applying the transformation yields the following points in $3D$:

$$D' = \begin{bmatrix} 1 & 1 \\ 0 & 2 \\ 0 & 2 \end{bmatrix}$$

In order to find the *SVM* decision boundary, we'll use the following formula which represents an optimization problem with respect to the constraints given in the problem description:

$$L_D(\alpha) = \sum_{i=1}^{2} \alpha_i - \frac{1}{2} \sum_{i=1}^{2} \sum_{i=1}^{2} \alpha_i \alpha_j z_i z_j x_i.x_j \tag{3.2}$$

This can be written as

$$L_D(\alpha) = \alpha_1 + \alpha_2 - \frac{1}{2}(\alpha_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \alpha_2 \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}).(\alpha_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \alpha_2 \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix})$$

simplifying the results yields in

$$L_D(\alpha) = \frac{-9}{2}\alpha_2^2 - \frac{1}{2}\alpha_1^2 + \alpha_1\alpha_2 + \alpha_1 + \alpha_2 \tag{3.3}$$

To solve (3.3), we should find the derivatives with respect to $\alpha_1$ and $\alpha_2$.

$$\frac{\partial L_D(\alpha)}{\partial \alpha_1} = -\alpha_1 + \alpha_2 + 1 = 0$$

$$\frac{\partial L_D(\alpha)}{\partial \alpha_2} = -9\alpha_2 + \alpha_1 + 1 = 0$$

Thus $\alpha_1 = \frac{5}{4}$ and $\alpha_2 = \frac{1}{4}$. Weights can be obtained using

$$W = \sum_{i=1}^{2} z_i x_i \alpha_i \tag{3.4}$$

Thus

$$W = \frac{5}{4}\begin{bmatrix}1\\0\\0\end{bmatrix} - \frac{1}{4}\begin{bmatrix}1\\2\\2\end{bmatrix} = \begin{bmatrix}1\\-1/2\\-1/2\end{bmatrix}$$

An obvious vector parallel with the weight vector is the normalized vector of W:

$$V = \begin{bmatrix}\frac{1}{\sqrt{1+\frac{1}{4}+\frac{1}{4}}}\\ \frac{\frac{-1}{2}}{\sqrt{1+\frac{1}{4}+\frac{1}{4}}}\\ \frac{\frac{-1}{2}}{\sqrt{1+\frac{1}{4}+\frac{1}{4}}}\end{bmatrix} = \begin{bmatrix}0.81\\-0.40\\-0.40\end{bmatrix}$$

The value of margin can be computed using following formula:

$$|M| = \frac{2}{|W|}$$

Replacing the results from section (a) yields the following results:

$$|M| = \frac{2}{\sqrt{1+1/4+1/4}} = 1.63$$

# 4 Hierarchical Clustring

Let $X = \{x_i,\ i = 1,2,3,4,5\}$, with $x_1 = [1,\ 1]^T$, $x_2 = [2,\ 1]^T$, $x_3 = [5,\ 4]^T$, $x_4 = [6,\ 5]^T$ and $x_5 = [6.5,\ 6]^T$. The pattern matrix of X is

$$D(X) = \begin{bmatrix}1 & 1\\2 & 1\\5 & 4\\6 & 5\\6.5 & 6\end{bmatrix}$$

Use the following methods to cluster the pattern matrix and draw the dendrogram for hierarchical models:

(a) Single Linkage

(b) Complete Linkage

(c) Average Linkage

(d) Mean Method

## Solution

(a) Note that in we've used *Manhattan* distance in all sections for simplicity. We'll fill our distance table with proper distances:

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| $x_1$ | 0 | 1 | 7 | 9 | 10.5 |
| $x_2$ | 1 | 0 | 6 | 8 | 9.5 |
| $x_3$ | 7 | 6 | 0 | 2 | 3.5 |
| $x_4$ | 9 | 8 | 2 | 0 | 1.5 |
| $x_5$ | 10.5 | 9.5 | 3.5 | 1.5 | 0 |

The *Single Linkage* method initializes each of the points in the dataset to a cluster respectively (a bottom-up approach).

### Step 1 (d = 1)

At the very first beginning, the lowest distance between two points is selected from the table. The cluster $\{x_1, x_2\}$ is made in this step.

### Step 2 (d = 1.5)

| | $\{x_1, x_2\}$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|
| $\{x_1, x_2\}$ | 0 | 6 | 8 | 9.5 |
| $x_3$ | 6 | 0 | 2 | 3.5 |
| $x_4$ | 8 | 2 | 0 | 1.5 |
| $x_5$ | 9.5 | 3.5 | 1.5 | 0 |

In this step, $x_4$ and $x_5$ are merged together with a distance of $d = 1.5$:

| | $\{x_1, x_2\}$ | $x_3$ | $\{x_4, x_5\}$ |
|---|---|---|---|
| $\{x_1, x_2\}$ | 0 | 6 | 8 |
| $x_3$ | 6 | 0 | 2 |
| $\{x_4, x_5\}$ | 9.5 | 2 | 0 |

### Step 3 (d = 2)

In step 3, $x_3$, $x_4$ and $x_5$ are being merged together.

| | $\{x_1, x_2\}$ | $\{x_3, x_4, x_5\}$ |
|---|---|---|
| $\{x_1, x_2\}$ | 0 | 6 |
| $\{x_3, x_4, x_5\}$ | 6 | 0 |

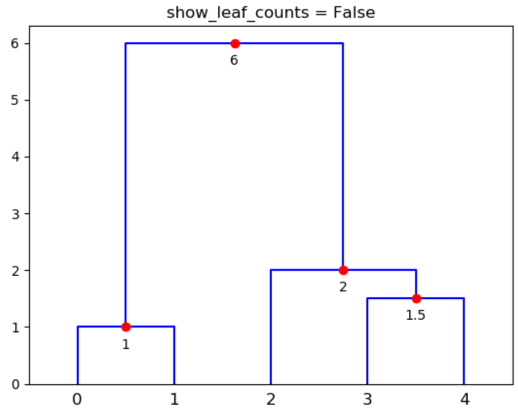


Figure 4.1: Dendrogram illustration of hierarchical clustering with *single link* method.

### Step 4 (d = 6)

In the final step, 2 clusters are merged together with $d = 6$. *Dendrogram* for this method is given in figure (4.1). (b) The difference between *Average Linkage* and *Single Linkage* method is **when** the clustering happens. In the *Complete Linkage* method, the average distance between two clusters are considered while merging two clusters. Following the same pattern for this section yields in the following results:

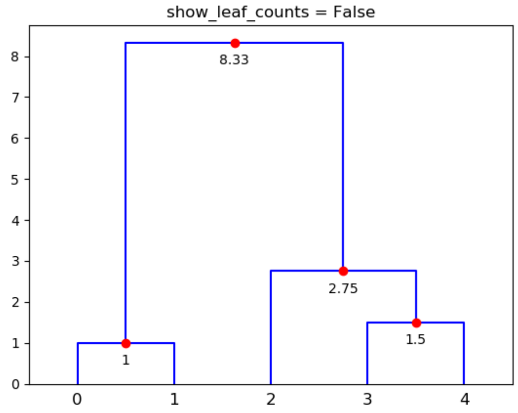


Figure 4.2: Dendrogram illustration of hierarchical clustering with *average link* method.

(c) The difference between *Complete Linkage* and *Single Linkage* method is **when** the clustering happens. In the *Complete Linkage* method, the maximum distance between two clusters are considered while merging two clusters. Following the same pattern for this section yields in the following results: The code for these sections is provided in the *src* folder. (d) In this case, the metric will be changed to *Euclidean*. The mean of each cluster

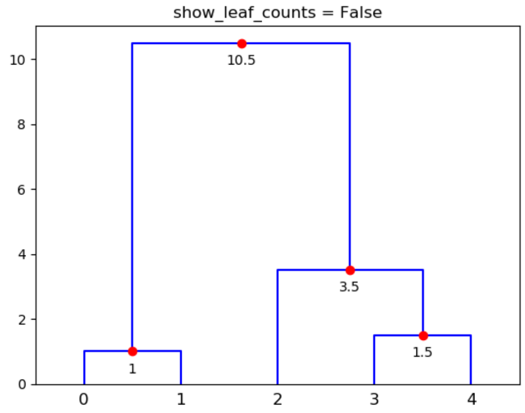


Figure 4.3: Dendrogram illustration of hierarchical clustering with *complete link* method.

is considered in each step and the distance of two clusters will be equal to the distance of *centroids* of two clusters. The resulting dendrogram is given in figure 4.4.

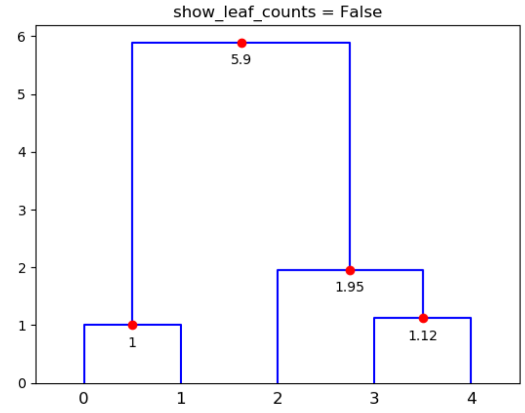


Figure 4.4: Dendrogram illustration of hierarchical clustering with *mean* method.