Assignment #4

* Adwaith Venkataraman (Andrew ID: adwaithv)

**Question 1**

**Analysis:**

1. The Euclidean distance is similar to the generic cartesian plane distance formula between (xi,yi) and (xj,yj) as .
2. The Cosine Angle coefficient is given by
3. The difference between the two methods are tabulated as follows:

|  |  |
| --- | --- |
| **Cosine Angle Coefficient** | **Euclidean Distance** |
| For similar inputs, the resultant value tends towards unity. | For similar inputs, the resultant value tends to nil. |
| The differences in orientation between the inputs are provided. | The differences in orientation between the inputs are not depicted. |
| The range of the output can share negative and positive values and is bounded between [-1,1] | The range of the output can only be in the positive domain of real numbers. |

**Results:**

When the results are computed for different input values, say non-negative inputs and negative inputs for the two cases, we get the results are follows:

|  |  |
| --- | --- |
| Case1: Non-negative Inputs | Case2: Negative Inputs Included |
| Feed array X: [1 1 1]  Feed array Y: [2 2 2]  euclidean\_dist =  1.7321  cosine\_angle\_coefficient =  1.0000 | Feed array X: [1 1 1]  Feed array Y: [-2 -2 -2]  euclidean\_dist =  5.1962  cosine\_angle\_coefficient =  -1.0000 |

**Matlab Code:**

x = input('Feed array X: ');

y = input('Feed array Y: ');

distance = 0;

numerator = 0;

denominator1 = 0;

denominator2 = 0;

if length(x)<length(y) %to take the shorter length arrays between the two

l=length(x);

elseif length(x)>length(y)

l=length(y);

else

l=length(x);

end

for i = 1:l

distance = distance + (x(i) - y(i))^2; %finding the square of the distance between the two array points

end

euclidean\_dist = sqrt(distance); %taking square root of the 'distance', we arrive at the euclidean distance

for i = 1:l

numerator = numerator + (x(i) \* y(i)); %the numerator is a cumulative addition of the product of corresponding elemts of the given arrays

denominator1 = denominator1 + x(i)^2; %the denominators are seperately calculated from the array elements

denominator2 = denominator2 + y(i)^2;

end

cosine\_angle\_coefficient = numerator / (sqrt(denominator1) \* sqrt(denominator2)); %the cosine angle coefficient is calculated

display(euclidean\_dist);

display(cosine\_angle\_coefficient);

**Question 2**

**Analysis:**

1. The given biometric features for the human subjects A,B and C is as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **A** | **B** | **C** |
| **1** | 1 | 1 | 1 |
| **2** | 1 | 1 | 0 |
| **3** | 0 | 0 | 1 |
| **4** | 0 | 0 | 1 |
| **5** | 1 | 1 | 0 |

1. The difference between the features of the given human subjects can be calculated by the following formula
2. The resultant is then multiplied by 100 in order to get the percentage difference between the two subject’ features.

**Results:**

The differences between the given subjects are tabulated as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **A** | **B** | **C** |
| **A** | N/A |  |  |
| **B** |  | N/A |  |
| **C** |  |  | N/A |

The percentage of difference is given as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **A** | **B** | **C** |
| **A** | 0% | 0% | 80% |
| **B** | 0% | 0% | 80% |
| **C** | 80% | 80% | 0% |

**Question 3**

**Analysis:**

1. Let us define the vectorial representation for the shapes as follows,

V0

V1

V2

V3

V4

V5

V6

V7

1. Now, defining the default vectorial combination of a car, truck and van, we get the following:

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Representation of a Car | Representation of a Truck | Representation of a Van |

1. The chain code for each of the vehicles can be given as:
   * Car: V2-V0-V1-V0-V7-V0-V6-V4
   * Truck: V2-V0-V2-V0-V6-V4
   * Van: V2-V0-V1-V0-V7-V6-V4
2. The Levenshtein distances can be computed by measuring the difference in the chain codes for each representation. The distances computed for the possible combinations are given as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Car** | **Truck** | **Van** |
| **Car** | Replacements:0  Deletions:0  Dist(C,C):0 | Replacements:3  Deletions:2  Dist(C,T):5 | Replacements:2  Deletions:1  Dist(C,V):3 |
| **Truck** | Replacements:3  Deletions:2  Dist(T,C):5 | Replacements:0  Deletions:0  Dist(T,T):0 | Replacements:3  Deletions:1  Dist(T,V):4 |
| **Van** | Replacements:2  Deletions:1  Dist(V,C):3 | Replacements:3  Deletions:1  Dist(V,T):4 | Replacements:0  Deletions:0  Dist(V,V):0 |

**Results:**

Considering a shape as shown below and analyzing with the default chain codes of the car, truck and the van, we arrive at the classification results.

|  |
| --- |
|  |
| New Vehicle Chain Code: V2-V0-V2-V0-V7-V6-V4 |

1. Comparing the chain code of the above vehicle with those of the default ones, we get

|  |  |  |  |
| --- | --- | --- | --- |
|  | Car | Truck | Van |
| New Vehicle | Replacements:3  Deletions:1  Dist(N,C):4 | Replacements:2  Deletions:1  Dist(N,T):3 | Replacements:1  Deletions:0  Dist(N,V):1 |

1. If the threshold for Dist(a,b) is set to 2, then the given vehicle will be classified as a Van, since the Dist(N,V)<Distthreshold.

**Question 4**

**Analysis:**

1. Let us consider the same outline of the objects as done in the previous question. However, we split the continuous chain into several fundamental components.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Representation of a Car | Representation of a Truck | Representation of a Van |

1. The chain code for each of the vehicles can be given as:
   * Car: V2-V0-V1- V0-V0-V7-V0-V6-V4-V4-V4-V4-V4-V4
   * Truck: V2-V0-V2-V0-V0-V0-V6-V6-V4-V4-V4-V4
   * Van: V2-V0-V1-V0-V0-V7-V6-V4-V4-V4-V4-V4
2. Now, repeating the same procedure, the Levenshtein distances are computed and shown as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Car** | **Truck** | **Van** |
| **Car** | Replacements:0  Deletions:0  Dist(C,C):0 | Replacements:3  Deletions:2  Dist(C,T):5 | Replacements:2  Deletions:2  Dist(C,V):4 |
| **Truck** | Replacements:3  Deletions:2  Dist(T,C):5 | Replacements:0  Deletions:0  Dist(T,T):0 | Replacements:3  Deletions:0  Dist(T,V):3 |
| **Van** | Replacements:2  Deletions:2  Dist(V,C):4 | Replacements:3  Deletions:0  Dist(V,T):3 | Replacements:0  Deletions:0  Dist(V,V):0 |

1. To calculate the modified Levenshtein distances, we compute the length of the longer chain code and find the percentage with respect to the existing Levenshtein distance, i.e.,

**Results:**

Considering the same shape as the previous problem for the new vehicle

|  |
| --- |
|  |
| New Vehicle Chain Code: V2-V0-V2-V0-V0-V0-V7-V6-V4-V4-V4-V4-V4 |

1. Comparing the chain code of the above vehicle with those of the default ones, we get

|  |  |  |  |
| --- | --- | --- | --- |
|  | Car | Truck | Van |
| New Vehicle | Replacements:3  Deletions:1  Dist(N,C):4 | Replacements:1  Deletions:1  Dist(N,T):2 | Replacements:4  Deletions:1  Dist(N,V):5 |
|  | 28.57% | 16.67% | 41.67% |

1. Therefore, objects of larger sizes can be handled with greater precision and a stricter threshold can be provided.

**Question 5**

**Analysis:**

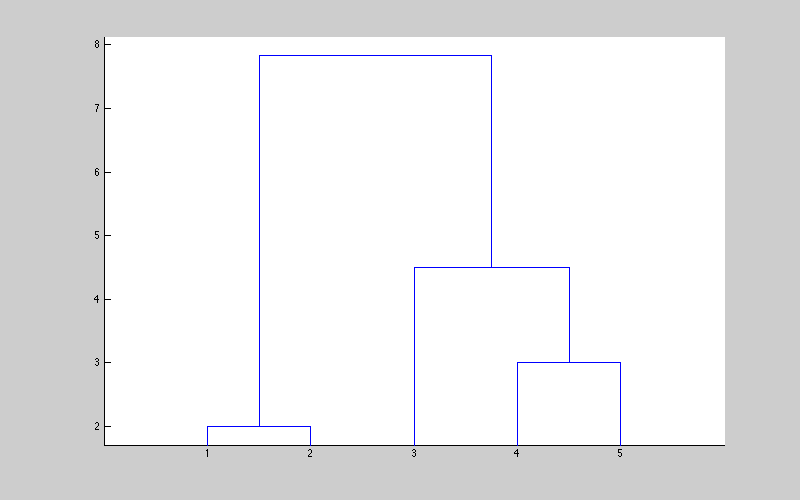
1. The given Euclidean matrix is as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **1** | **2** | **3** | **4** | **5** |
| **1** | 0 |  |  |  |  |
| **2** | 2 | 0 |  |  |  |
| **3** | 6 | 5 | 0 |  |  |
| **4** | 10 | 9 | 4 | 0 |  |
| **5** | 9 | 8 | 5 | 3 | 0 |

1. The given matrix is fed into a 1-D matrix.
2. This matrix is now fed to the linkage function of matlab that returns a matrix after encoding a tree of hierarchical cluster of the rows of the input matrix.
3. The linkage matrix is now fed to the dendrogram function that provides us with the resultant dendrogram.

**Results:**

The resultant dendrogram was obtained as shown:



**Matlab Code:**

a=[2, 6, 10, 9, 5, 9, 8, 4, 5, 3]; %the given values are fed as input

b=linkage(a,'average'); %the linkage matrix is formed

dendrogram(b,5) %the desired dendrogram is obtained

**Question 6**

**Analysis:**

1. The given linearized function can be mapped to form the Gaussian membership function by specifying the values of and c in the following equation,
2. The specified domain ranges from -3.0 to +6.0 with a uniform spacing of 1.0 between values.
3. The values of and c represent the gradient of the slope and the mean value of obtaining the Gaussian peak. Let us say, we specify values of 1 and 1 for and c respectively. We notice that the peak of the Gaussian membership function lies at X=1 and the gradient of the slope is 1.
4. Gaussian curves for various values are computed and tabulated.

**Results:**

|  |  |
| --- | --- |
|  |  |
| Gaussian mapping for and c=1 | Gaussian mapping for and c=1 |
|  |  |
| Gaussian mapping for and c=3 | Gaussian mapping for and c=3 |

**Matlab Code:**

x=-3:0.05:6; %Initializing the range of 'x' values

y = gaussmf(x, [3 3]); %plotting the guassian curve

plot(x,y);

xlabel('gaussmf, P=[3 3]')

ylabel('Mapped Values')

**Question 7**

**Analysis:**

1. Given that the two training samples are at X=0 and X=2, we would need to optimize the condition, by minimizing the value of,

and subject to the constraints,

In this case, the constraints can be given as,

1. From the above conditions and the constraints, we arrive at the following inequalities:

1. The minimum value that *w* can take is 1.
2. When applying the minimum value of *w* in the optimizing equation, we get

**Result:**

Hence, x=1/2 is the middle of the two training data.