Assignment #4

- Adwaith Venkataraman (Andrew ID: adwaithv)

Question 1

Analysis:

1. The Euclidean distance is similar to the generic cartesian plane distance formula between (x_i,y_i) and (x_j,y_j) as $D = \sqrt[2]{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}$.

2. The Cosine Angle coefficient is given by
$$Cosine\ Angle\ Coefficient = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt[2]{\sum_{i=1}^{n}(A_i)^2} * \sqrt[2]{\sum_{i=1}^{n}(B_i)^2}}$$

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

2. The difference between the features of the given human subjects can be calculated by the following formula

$$Difference\ between\ subject, S_i\ and\ subject, S_j, D_{ij} = 1 - \frac{M_{11}}{(M_{11} + M_{01} + M_{10})}$$

3. The resultant D_{ij} 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	$D_{AB} = 1 - \frac{3}{(3+0+0)} = 0$	$D_{AC} = 1 - \frac{1}{(1+2+2)} = 0.8$
B	$D_{BA} = 1 - \frac{3}{(3+0+0)} = 0$	N/A	$D_{BC} = 1 - \frac{1}{(1+2+2)} = 0.8$
C	$D_{CA} = 1 - \frac{1}{(1+2+2)} = 0.8$	$D_{CB} = 1 - \frac{1}{(1+2+2)} = 0.8$	N/A

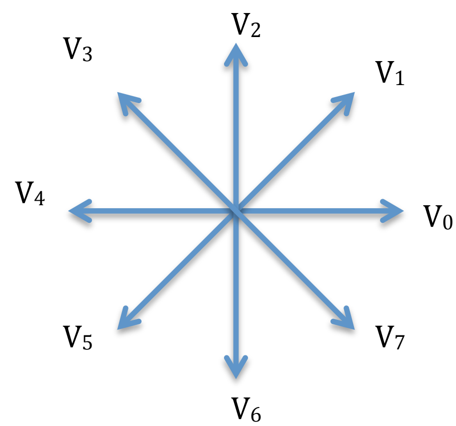
The percentage of difference is given as follows:

	A	**B**	**C**
A	0%	0%	80%
B	0%	0%	80%
C	80%	80%	0%

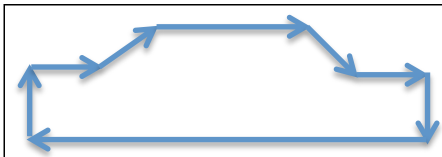
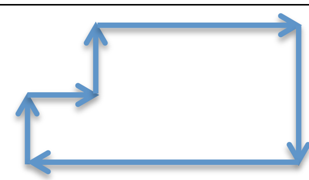
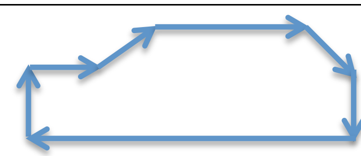
Question 3

Analysis:

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



2. 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

3. The chain code for each of the vehicles can be given as:
   - Car: V_2-V_0-V_1-V_0-V_7-V_0-V_6-V_4
   - Truck: V_2-V_0-V_2-V_0-V_6-V_4
   - Van: V_2-V_0-V_1-V_0-V_7-V_6-V_4

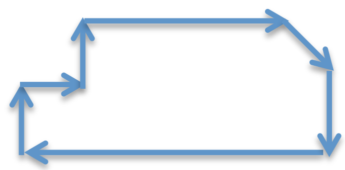
4. The Levenshtein distances can be computed by measuring the difference in the chain codes for each representation. The distances computed for the possible combinations are given as follows:

	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	Replacements:3 Deletions:1	Replacements:0 Deletions:0

	Dist(V,C):3	Dist(V,T):4	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: V_2-V_0-V_2-V_0-V_7-V_6-V_4

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

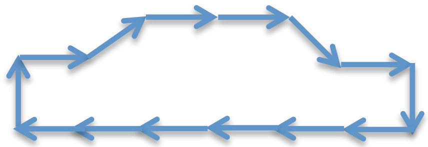
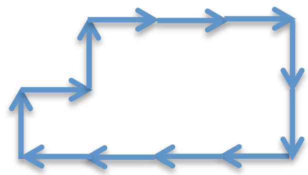
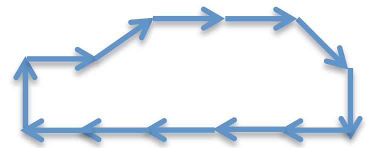
	Car	Truck	Van
New Vehicle	Replacements:3 Deletions:1 Dist(N,C):4	Replacements:2 Deletions:1 Dist(N,T):3	Replacements:1 Deletions:0 Dist(N,V):1

6. 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)<$Distt_{hreshold}$.

<u>Question 4</u>

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

2. The chain code for each of the vehicles can be given as:
   - Car: V_2-V_0-V_1- V_0-V_0-V_7-V_0-V_6-V_4-V_4-V_4-V_4-V_4-V_4
   - Truck: V_2-V_0-V_2-V_0-V_0-V_0-V_6-V_6-V_4-V_4-V_4-V_4
   - Van: V_2-V_0-V_1-V_0-V_0-V_7-V_6-V_4-V_4-V_4-V_4-V_4

3. Now, repeating the same procedure, the Levenshtein distances are computed and shown as follows:

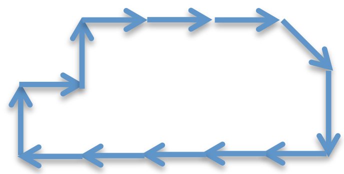
	Car	**Truck**	**Van**
Car	Replacements:0 Deletions:0 Dist(C,C):0	Replacements:3 Deletions:2 Dist(C,T):5	Replacements:2 Deletions:2 Dist(C,V):4
Truck	Replacements:3 Deletions:2 Dist(T,C):5	Replacements:0 Deletions:0 Dist(T,T):0	Replacements:3 Deletions:0 Dist(T,V):3
Van	Replacements:2 Deletions:2 Dist(V,C):4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

4. 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.,

$$modified\ Levenshtein\ distance, L_d^m = \frac{L_d}{Length\ of\ Item\ with\ longer\ chain\ code} * 100$$

Results:

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



New Vehicle Chain Code: V_2-V_0-V_2-V_0-V_0-V_0-V_7-V_6-V_4-V_4-V_4-V_4-V_4

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
L_d^m	28.57%	16.67%	41.67%

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

Question 5

Analysis:

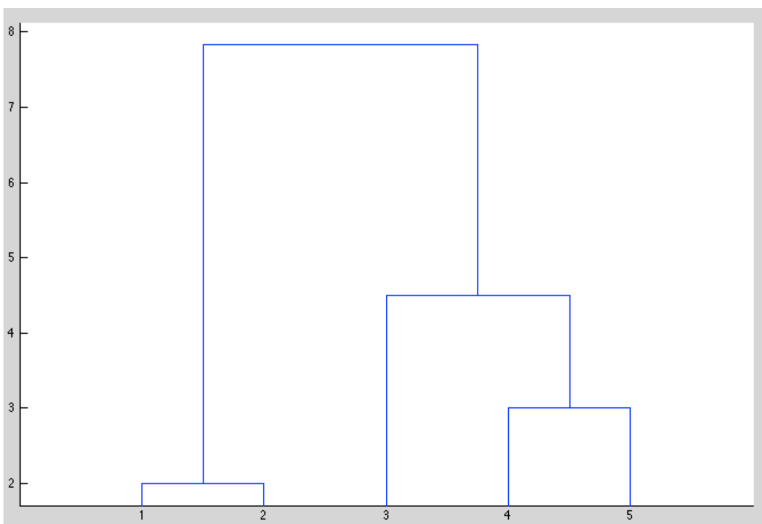
1. The given Euclidean matrix is as follows:

	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

2. The given matrix is fed into a 1-D matrix.
3. 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.
4. 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:

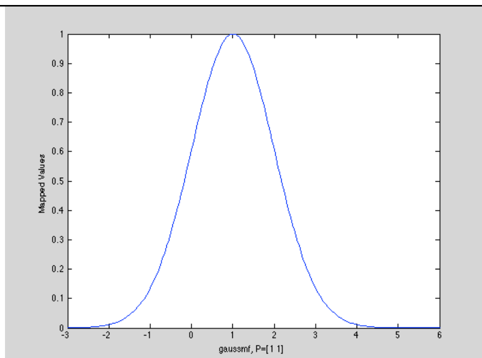
```
a=[2, 6, 10, 9, 5, 9, 8, 4, 5, 3]; %the given values are fed as input
b=linkage(a,'average'); %the linkage matrix is formed
dendrogram(b,5) %the desired dendrogram is obtained
```

<u>**Question 6**</u>

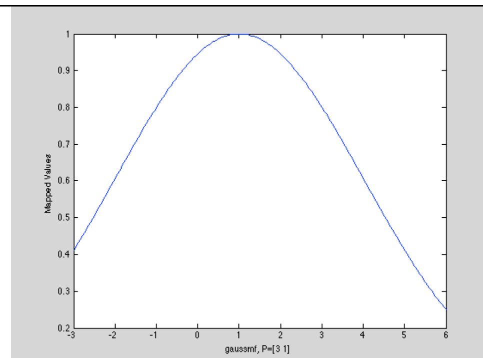
Analysis:

1. The given linearized function can be mapped to form the Gaussian membership function by specifying the values of σ and c in the following equation,
$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$
2. The specified domain ranges from -3.0 to +6.0 with a uniform spacing of 1.0 between values.
3. The values of σ 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 σ 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.

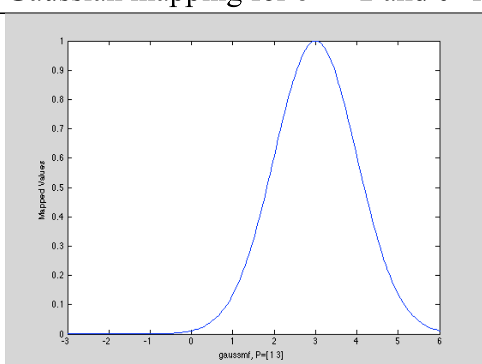
Results:



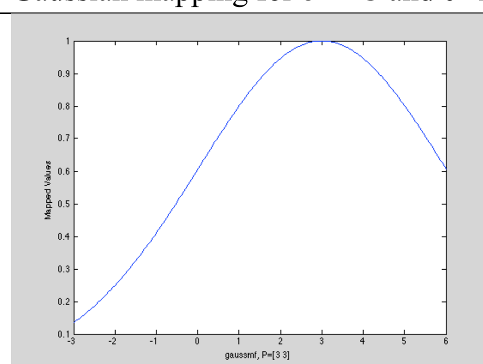
Gaussian mapping for $\sigma = 1$ and c=1



Gaussian mapping for $\sigma = 3$ and c=1



Gaussian mapping for $\sigma = 1$ and c=3



Gaussian mapping for $\sigma = 3$ 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')
```

<u>Question 7</u>

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,

$$\frac{1}{2}w^t w$$
$$= \frac{1}{2}w^2$$

   and subject to the constraints,

$$w^t x_i + b \geq 1 \; and \; w^t x_i + b \leq -1$$

   In this case, the constraints can be given as,

$$w.2 + b \geq 1 \; and \; w.0 + b \leq -1$$

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

$$2w \geq 1 - b;$$
$$b \leq -1;$$
$$Implying, -b \geq 1 \; and \; hence \; 2w \geq 2,$$
$$Therefore, w \geq 1$$

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

$$x = \frac{1}{2}w^2$$
$$= \frac{1}{2}1^2$$
$$x = \frac{1}{2}$$

Result:

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