Öğrenci Numarası: 200517005

İsim-Soyisim: Cemil GÜNDÜZ

Ödevin Konusu: Bilgisayar Programlama Dersi Ödev 1

**İÇİNDEKİLER DİZİNİ**

İçindekiler Tablosu

**Bölüm başlığını yazın (düzey 1)1**

Bölüm başlığını yazın (düzey 2)2

Bölüm başlığını yazın (düzey 3)3

**Bölüm başlığını yazın (düzey 1)4**

Bölüm başlığını yazın (düzey 2)5

Bölüm başlığını yazın (düzey 3)6

**Şekiller Dizini**

Şekil tablosu öğesi bulunamadı.

**Ödev Cevapları**

1. **Soru:**

Bu soru, basit bir doğal dil işleme ve sınıflandırma uygulamasıdır. Verilen bir örneğin hangi gruba dahil olduğunun belirlenmesi amaçlanmaktadır. Bunun için öncelikle haber metinlerinden belli kriterler doğrultusunda nitelik çıkarımı gerçekleştirilmiştir. Bu nitelikler metindeki harf ve kelimelerin işlenmesi sonucunda elde edilmektedir.

Sınıflandırma öncesinde iki farklı haber sitesinden metinler elde edilmiş, bu metinler news klasöründe her bir haber bir txt dosyasında olacak şekilde kaydedilmiştir. Her bir haber sitesinden 10 ar haber ve bir de üçüncü bir haber sitesinden test edilmek üzere haber kaydedilmiş, toplamda 21 haber mevcuttur.

Her bir gruptaki 10’ar haber metninden 15’er nitelik çıkarılarak 10\*15’lik 2 matris oluşturulmuştur. Ardından test için kullanılacak metinden 1\*15’lik bir vektör elde edilmiş ve bu vektörün hangi gruba daha yakın olduğunun tespiti için öklit ve mahalanobis uzaklıkları hesaplanmıştır. Bu uzaklıklar yazdırılarak hangi gruba yakın olduğu belirlenmesi sağlanmıştır.

group1 = getNewsGroup("sozcu");

group2 = getNewsGroup("sabah");

test = calculateVector(fileread("news\hurriyet.txt"));

euclidean\_distance\_to\_news\_group1 = calculateEuclid(group1, test)

euclidean\_distance\_to\_news\_group2 = calculateEuclid(group2, test)

mahalanobis\_distance\_to\_news\_group1 = calculateMahal(group1, test)

mahalanobis\_distance\_to\_news\_group2 = calculateMahal(group2, test)

Yukarıdaki kod bloğunda, txt dosyalarından metinler okunup matrisler ve test vektörü aşağıdaki fonksiyonlar yardımı ile hazırlanmıştır.

function euclid = calculateEuclid(news\_matrix, test)

distances = zeros(10,1);

for i = 1:size(news\_matrix,1)

distances(i) = sqrt(sum((news\_matrix(i,:)- test) .^2));

end

euclid = mean(distances);

end

Kümeleme uygulamalarında bir örneğin hangi gruba yakın olduğunun belirlenmesinde farklı yöntemler uygulanabilir. Test edilen örneğe en yakın olan örnek hangisi ise onun grubuna dahil etmek veya gruplar içerisinde en uzak örneğin en yakınının olduğu gruba dahil etmek veya grup örneklerinin uzaklıkları ortalamasının en yakınına dahil etmek gibi... Yukarıdaki fonksiyonda iki grubun da tüm elemanlarının yeni örneğe uzaklıkları hesaplanmıştır. Yeni örnek uzaklıklar ortalaması en yakın olan gruba dahil edilebilir.

function mahalanobis = calculateMahal(news\_matrix, test)

covX = cov(news\_matrix);

mu = mean(news\_matrix,1);

inX = inv(covX);

mahalanobis = sqrt((test - mu) \* inX \* (test-mu)');

end

Bu metot, parametre olarak haber grubunun matrisini ve test vektörünü alıp test vektörünün bu matristeki elemanlara mahalanobis uzaklığını vermektedir. Bunun için haber matrisinin covaryansı ve matris elemanlarının ortalamaları hesaplanır. Sonrasında test örneğinin grup ortalamalarına öklit uzaklığı hesaplanarak mahalanobis uzaklığı bulunabilir.

function group\_matrix = getNewsGroup(news)

group\_matrix = [];

for i=1:10

news\_text = fileread("news\" + string(news) + i + ".txt");

news\_matrix = calculateVector(news\_text);

group\_matrix = [group\_matrix;news\_matrix];

end

end

Bu metot her bir haber grubundan metinlerin okunup her bir haber metninin vektörlerinin hesaplatan sonrasında da bu vektörleri news\_matrix değişkenlerine aktaran bir yardımcı metotdur.

function result\_vector = calculateVector(text)

text = char(text);

%kelime bigram/trigramlarında kullanılmak üzere basit metin temizleme yapıldı.

%kelime sayısı, ortalama harf sayısında da temizlenmiş metin kullanıldı.

%ltext = temizlenmiş metin

ltext = lower(text);

ltext = erasePunctuation(ltext);

words = string(split(ltext, ' '));

word\_count = length(words);

avg\_letter\_count = mean(strlength(words));

uppercase\_count = sum(isstrprop(text, "upper"));

lovercase\_count = sum(isstrprop(text, "lower"));

digit\_count = sum(isstrprop(text, "digit"));

punctuation\_count = sum(isstrprop(text, "punct"));

special\_count = getSpecialCharCount(text);

char\_uppercase\_bigrams = getBiGramCount(text, 'upper');

char\_lowercase\_bigrams = getBiGramCount(text, 'lower');

char\_uppercase\_trigrams = getTriGramCount(text, 'upper');

char\_digit\_bigrams = getBiGramCount(text, 'digit');

char\_digit\_trigrams = getTriGramCount(text, 'digit');

char\_lowercase\_trigrams = getTriGramCount(text, 'lower');

word\_bigrams = getWordBiGram(words);

word\_trigram = getWordTriGram(words);

result\_vector = [word\_count avg\_letter\_count uppercase\_count lovercase\_count digit\_count punctuation\_count special\_count char\_uppercase\_bigrams char\_lowercase\_bigrams char\_uppercase\_trigrams char\_digit\_bigrams char\_digit\_trigrams char\_lowercase\_trigrams word\_bigrams word\_trigram];

end

# Bu metot metinlerden nitelik vektörlerinin çıkarılmasında kullanılır. Bazı niteliklerin hesaplanmasında kullanılmak üzere basit metin temizleme de uygulanmıştır. Buradaki amaç, noktalama işaretleri ve büyük küçük harf gibi durumların bigram-trigram hesaplamalarını engellemesinin önüne geçmektir. Aksi halde “kalem,” “Kalem” gibi kelimeler iki farklı ifadeymiş gibi değerlendirilecektir.

function c\_spec = getSpecialCharCount(text)

text = text(isstrprop(text, 'alpha'));%sadece letters içerisinde özel karakter arasın..

d\_text = double(text);

c\_spec = sum(d\_text>191); %UNICODE'da 192ve yukarısı özel karakterler olduğundan..

end

Bu metot, parametre olarak verilen metindeki özel karakter sayısını belirler. İşlem karmaşıklığını düşürmek için öncelikle isstrprop fonksiyonundan yararlanarak sadece harfler elde edilmiş, ardından bu yeni metin içinde özel karakterler aranmıştır. Bunun için karakterlerin double değerleri alınmış ve Unicode kodlamasında 192 sonrasında yer alan özel karakterler arasında olup olmadığı kontrol edilmiştir.

function char\_BG = getBiGramCount(text, category)

char\_BG =0;

for i=1:length(text)-1

if isstrprop(text(i), category)==1

if text(i) == text(i+1)

char\_BG = char\_BG +1;

end

end

end

end

Bu metot, metindeki büyük, küçük ve rakamlar ikililerini döndürür. Metot, iki parametre alır, ilki ikililerin aranacağı metin, diğeri de isstrprop metodunda da kullanılan category parametresidir. Bu parametreye gönderilen ‘upper’, ‘lower’ ve ‘digit’ ifadeleri hangi türde ikililerin aranacağını ifade eder.

function char\_TRG = getTriGramCount(text, category)

char\_TRG = 0;

for i = 2 : length(text)-2

if isstrprop(text(i), category)==1

if text(i-1) ==text(i) && text(i+1) ==text(i)

char\_TRG = char\_TRG +1;

end

end

end

end

Yukarıdaki fonksiyonun üçlüleri arayan versiyonudur. Bir harfin öncesi ve sonrasındaki ile aynı olup olmadığını kontrol eder. Aynı ise sayıya ekler.

function word\_BG = getWordBiGram(word\_vector)

word\_BG = 0;

for i = 1: length(word\_vector)-1

if word\_vector(i) == word\_vector(i+1)

word\_BG = word\_BG +1;

end

end

end

Bu fonksiyon, kelime ikililerini tespit etmede kullanılır. Bunun için temizlenmiş metin boşluklara göre bölünmüş ve bir kelime vektörü elde edilmiştir. Vektör elemanları sıralı olarak gezinilerek arka arkaya aynı kelimeler varsa sayıya eklenmiştir.

function word\_TRG = getWordTriGram(word\_vector)

word\_TRG = 0;

for i = 2: length(word\_vector)-2

if word\_vector(i) == word\_vector(i+1) && word\_vector(i) == word\_vector(i-1)

word\_TRG = word\_TRG +1;

end

end

end

Bu fonksiyon kelime üçlülerini tespit etmek için kullanılır. Benzer şekilde kelime vektöründen yararlanılır. Önceki ve sonraki kelime ile aynı olan elemanlar sayıya dahil edilir.

function new\_text = erasePunctuation(ltext)

ltext = replace(ltext,',',' ');

ltext = replace(ltext,'.',' ');

ltext = replace(ltext,'?',' ');

ltext = replace(ltext,'!',' ');

ltext = replace(ltext,'''',' ');

ltext = replace(ltext,'-',' ');

ltext = replace(ltext,')',' ');

ltext = replace(ltext,'(',' ');

ltext = replace(ltext,';',' ');

ltext = replace(ltext,':',' ');

ltext = replace(ltext,'’',' ');

ltext = regexprep(ltext, '\s+',' ');

ltext = replace(ltext,' ',' ');

ltext = replace(ltext,' ',' ');

new\_text = ltext;

end

Metin temizlemede kullanılan yardımcı fonksiyondur. Metindeki noktalama işaretlerini ve varsa birden fazla boşlukları temizler.

1. **Soru:**

**Bu soruda 200\*200’lük bir matriste öğrenci numarasına göre belirlenmiş başlangıç ve bitiş noktaları arasında öklit uzaklıkları belirlenmesi istenmiştir. Buna göre her öncelikle öğrenci numarası ve isme göre başlangıç ve bitiş noktaları belirlenecek, ardından başlangıç ve bitiş noktaları arasında her bir satırdaki noktaların bitiş noktasına öklit uzaklıkları hesaplanacaktır. Hesaplanan öklit uzaklıklarında her bir satırın en yakın elemanı 1 diğer noktalar 0 olarak işaretlenecektir. Ek olarak bazı engellenmiş noktalar belirlenmesi ve bu noktalarda uzaklık en yakın olsa bile bu noktadan geçilmemesi istenmiştir. Bu doğrultuda işlemler yapılmıştır.**

studentID = '200517005';

student\_name = 'Cemil';

initial\_matrix = zeros(200,200);

Hesaplamada kullanılacak değişkenlerin tanımlanması yapılmıştır.

[st\_row, st\_col] = calculateStart(studentID, student\_name);

[fin\_row, fin\_col] = calculateFinish(student\_name);

Yardımcı fonksyionlar yardımı ile başlangıç ve bitiş noktaları hesaplanmıştır. Bitiş noktasındaki

%this part is for the restriction of the robot.

%if restricted then checks if robot is on a restricted cell

restricted\_points = [18 71; 19 71; 19 70; 20 71; 20 70; 20 69 ];

restrict\_robot = true;

Bu blok robotun kısıtlanması ile ilgilidir. Belirlenen noktalar en yakın nokta olsa dahi bu noktaların işaretlenmesinin önüne geçilmiştir. Boolean bir değişken yardımı ile kısıtlama işleminin açılıp kapatılması da sağlanmıştır.

for i=st\_row:fin\_row

row\_mins = zeros(1,fin\_col-st\_col+1);

k=1;

for j=st\_col:fin\_col

dist = sqrt((fin\_row-i)^2 + (fin\_col-j)^2);

if restrict\_robot && ismember([i j], restricted\_points, "rows")

%if robot is restricted and is on a restricted cell

%then set distance to max so it wont move on this cell

dist = 200\*200;

end

row\_mins(k)= dist;

initial\_matrix(i,j)=0;

k=k+1;

end

[M, ind] = min(row\_mins);

initial\_matrix(i,st\_col+ind-1) = 1;

end

imshow(initial\_matrix)

Bu kod bloğu asıl gezinimin yapıldığı kısımdır. Satır bazında her bir nokta taranmış, her bir noktanın bitiş noktasına uzaklıkları hesaplanmıştır. Eğer kısıtlama açıksa ve nokta kısıtlanmış noktalar arasında ise o noktanın uzaklığı olabilecek maksimum uzaklığa eşitlenmiştir.

function [st\_row, st\_col] = calculateStart(studentID, student\_name)

last\_two = str2double(studentID(end-1:end));

first\_char = double(student\_name(1));

st\_row = mod(last\_two,10);

st\_col = mod(first\_char,10);

end

Bu metot başlangıç noktası hesaplamada kullanılan bir yardımcı metottur.

function [fin\_row, fin\_col] = calculateFinish(student\_name)

first\_char = double(student\_name(1));

fin\_row = maxPrime(first\_char);

fin\_col = minPrime(first\_char);

end

Bu metot, bitiş noktasının hesaplamasında kullanılan yardımcı metottur.

function min\_prime = minPrime(n)

n=n+1;

while isPrime(n)==false

n=n+1;

end

min\_prime = n;

end

Bu metot, kendisine verilen n parametresinden büyük en küçük asal sayıyı hesaplamada kullanılır.

function max\_prime = maxPrime(n)

max\_prime = 2;

for i=3:2:n

if isPrime(i)==true

max\_prime = i;

end

end

end

Bu metot, bir sayının kendinden küçük en büyük asal sayıyı hesaplamada kullanılır. Asal sayı metotları bitiş noktasının hesaplanmasında kullanılmıştır.

function is\_prime = isPrime(n)

is\_prime = true;

for i=2:n-1

if mod(n,i)==0

is\_prime = false;

break

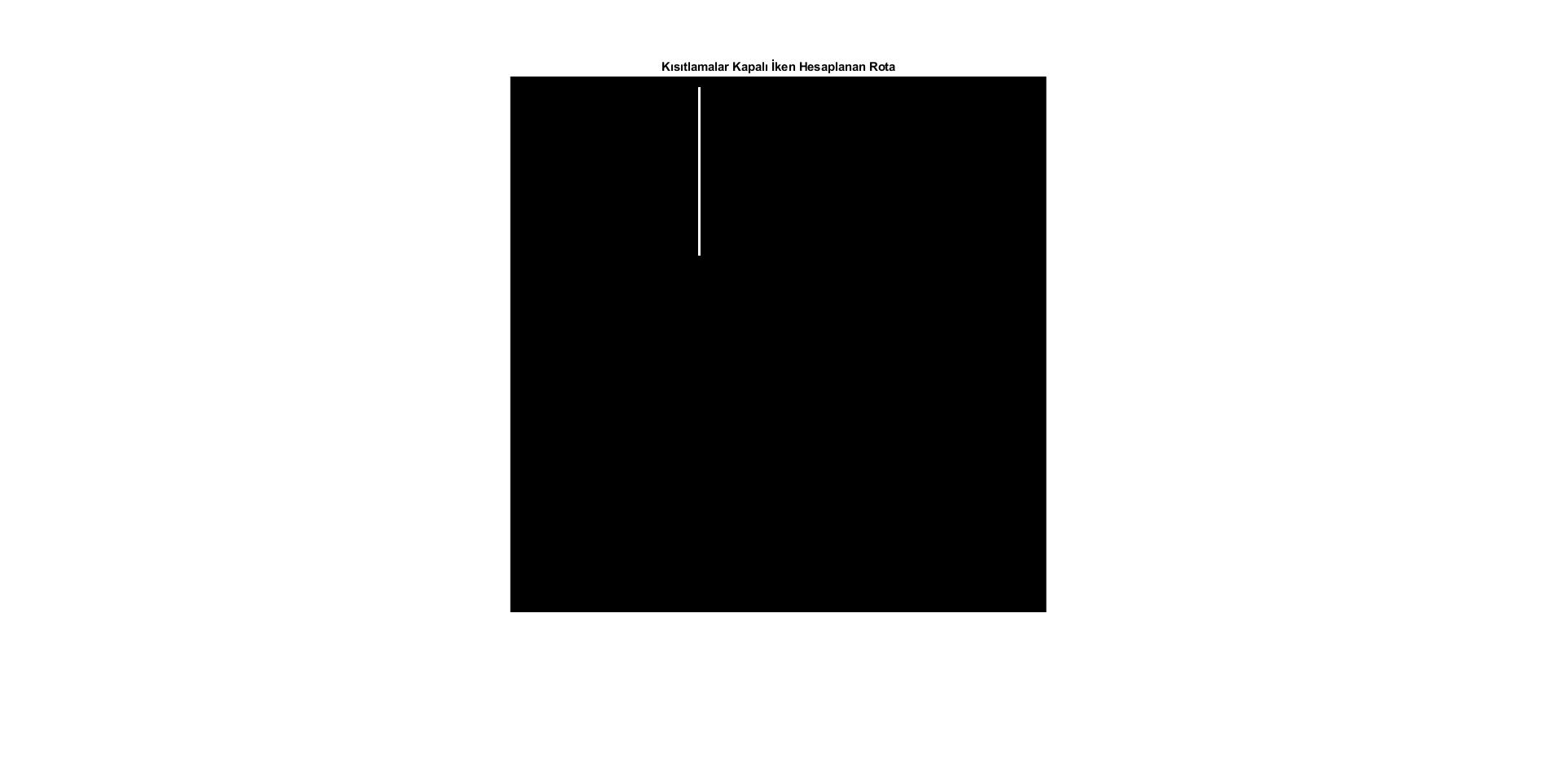
end

end

end

Bu metot kendisine verilen n parametresinin asal olup olmadığını belirlemede kullanılan bir yardımcı metottur. Asal sayı fonksiyonları, kodun okunabilirliğini artırma amacı ile yazılmıştır.

Kısıtlamalar açık ve kapalı olarak elde edilen sonuçlar aşağıdaki resimlerde sunulmuştur.



* **Soru ile ilgili açıklamalar**
* **Yazılan Kodlar Açıklaması ( Blok**

1. **Soru:**

* **Soru ile ilgili açıklamalar**
* **Yazılan Kodlar Açıklaması ( Blok Blok)**
* **Elde edilen Sonuçlar**

1. **Soru:**

**Bu soruda, verilen bir float değerin binary ifadesinin hesaplanması yapılmıştır. 32 bitlik binary ifadede ilk bit değerin işaretini, sonraki 8 bit exponansiyelini, 23 bitlik kalan kısım ise mantissa kısmını ifade etmektedir.**

**Hesaplama için öncelikle işaret belirlenir. Sayı pozitif ise 0, negatif ise 1 olarak belirlenir. Ardından sayının tamsayı kısmı alınarak tamsayının binary dönüşümü gerçekleştirilir. 3. Olarak sayının ondalık kısmı alınır ve sürekli olarak 2 ile çarpılır ve çarpmanın tamsayı kısmı not edilir. Not edilen tamsayı kısımlar baştan sona yazılarak ondalık kısmın binary ifadesi elde edilir.**

**2 ile çarpma işlemi sonucun ondalık değeri 0 olana kadar veya 24 sefer devam ettirilir. 23. Seferde de ondalık kısmın 0 olmaması demek, sayının bu dönüşümde hassaslık (precision) kaybettiği anlamına gelir.**

**4. adımda tamsayı ve ondalık kısımlar art arda yazılarak tüm sayının tamsayı kısmı elde edilir. Bu sayıda virgül kaydırma yapılır. Virgül kaç basamak kaydırılmış ise sayının exponansiyeli odur. Ardından bu ifadeye 127 eklenerek sonuç binary’ye dönüştürülür ve sayının exponansiyelinin binary ifadesi (8 bitlik kısım) elde edilir.**

**Mantissa, 4. Adımda elde edilen virgül kaydırılmış kısmın virgülden sonraki kısmıdır. Bu kısmın ilk 23 basamağı alınır, atılan kısımda 1 varsa yine hassaslık kaybı olduğu anlamına gelir.**

**Son olarak işaret, exponentin binary ifadesi ve mantissa birleştirilerek float değerin binary ifadesine erişilir.**

**Binary ifadeden hexadecimale dönüşüm için ise 4’erli gruplar halinde dönüşüm gerçekleştirilir. Dörtlü ifadelerin her biri, hexadecimaldeki bir basamakla ifade edilir.**

%num = -50.0001;

num = input('Ondalıklı bir sayı giriniz : ');

binary\_vector = floatToBinary(num);

binary\_representation = strjoin(binary\_vector, ' ')

hexadecimal = binaryToHex(floatToBinary(num));

hex\_representation = strjoin(['0 x ', string(hexadecimal)], '')

Bu kod bloğunda kullanıcıdan bir sayı alınmış ve bu sayı fonskyionlar yardımı ile önce binary’ye sonra binaryden hexadecimale dönüştürülmüştür.

function result = floatToBinary(num)

%this is the main function that calculates

%binary representation of float value

sign = double(not(num>0));

[intnum, dec] = splitNumber(num);

int\_binary = toBinary(intnum);

dec\_binary = toDecBinary(dec);

exponent\_val = length(int\_binary)-1;

exp\_binary = toBinary(exponent\_val+127);

mantissa = [int\_binary(2:end),dec\_binary];

if length(mantissa)>23

mantissa = mantissa(1:23); %if longer: take 23 digits

else

mantissa(23) = 0;%normalize mantissa to 23 digits

end

result = [sign, exp\_binary, string(mantissa)];

end

Bu fonksiyon, verilen float değerin binary gösterimini hesaplayan ana fonksiyondur. Yardımcı fonksiyonlar yardımı ile binary gösterimin bileşenleri hesaplanır ve bu fonksyion yardımı ile normalleştirilir.

function hex = binaryToHex(binary\_vector)

%this method calculates hexadecimal representation of a binary float

hex = strings(1,8);

bv = double(binary\_vector); % for readability

for i = 0:7

sumX = bv(i\*4+1)\*2^3 + bv(i\*4+2)\*2^2 + bv(i\*4+3)\*2^1 + bv(i\*4+4)\*2^0;

hex\_Values = ["A", "B", "C", "D", "E", "F"];

if sumX > 9

sumX = hex\_Values(mod(sumX, 10)+1);

end

hex(i+1) = string(sumX);

end

end

Bu metot, verilen binary gösterim üzerinden hexadecimal hesabını gerçekleştiren fonksiyondur. Parametre olarak verilen binary gösterimdeki 4’lü gruplar yardımı ile hexadecimal gösterim belirlenir.

function result = toBinary(n)

rem = [];

while n>1

remainder = mod(n,2);

division = fix(n/2);

rem(end+1) = remainder;

n=division;

end

rem(end+1) = 1;

result = flip(rem);

end

Bu fonksiyon verilen bir decimal değeri binary’ye dönüştürmede kullanılan bir yardımcı metottur.

function dec\_bin = toDecBinary(n)

cnt = 1;

intparts = [];

[~, dec] = splitNumber(n);

while cnt<25 && dec ~=0

[intnum, dec] = splitNumber(n\*2);

intparts = [intparts, intnum];

n = dec;

cnt=cnt+1;

end

dec\_bin = intparts;

end

Yukarıda verilen fonksiyon, 10’luk tabanda verilen bir ondalık değerin binary gösterimini hesaplamada kullanılır.

function [intnum, dec] = splitNumber(n)

abnum = abs(n);

intnum = fix(abnum); %tamsayı kısmı

dec = abnum-intnum; %decimal kısmı

end

Bu metot, parametre olarak verilen virgüllü sayının tamsayı ve ondalık kısımlarını ayırıp iki farklı parametre olarak döndüren bir yardımcı fonksiyondur.

Sonuçlar :

Bu soru yardımı ile float sayıların bilgisayarda binary olarak gösterimi öğrenilmiştir. Aynı zamanda kazanımlardan bir diğeri de, bilgisayarda bu dönüşüm esnasında float değerlerde bazı hassasiyet kayıpları olduğudur.

* **Soru ile ilgili açıklamalar**
* **Yazılan Kodlar Açıklaması ( Blok Blok)**
* **Elde edilen Sonuçlar**