گزارش پروژه دوم

درس انتقال داده ها

فصل سوم نظریه اطلاعات

نگین حقیقی – 99521226

استاد درس: دکتر ابوالفضل دیانت

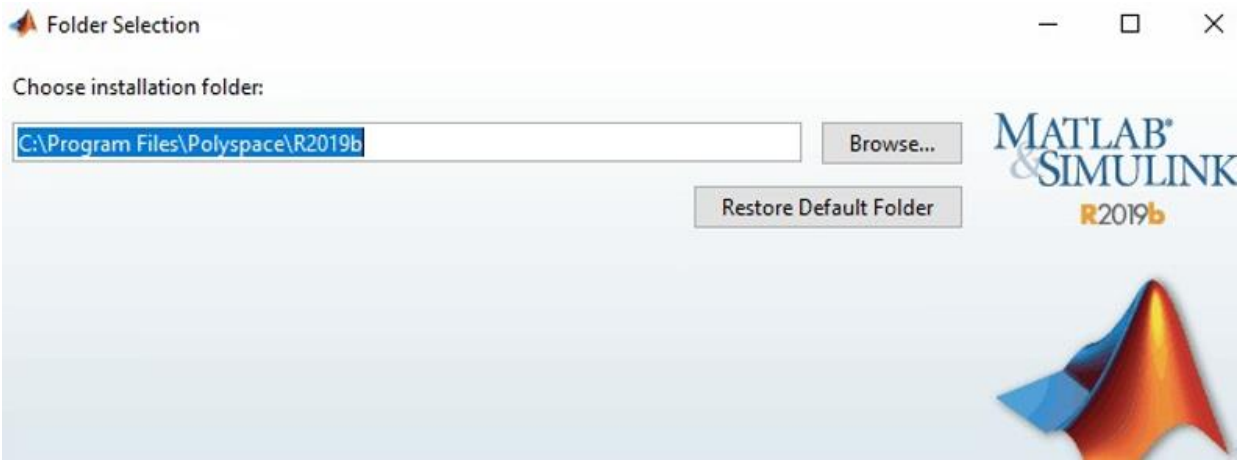
نیم سال دوم 1402-1401

مقدمه:

در این پروژه قصد داریم یک فایل صوتی را وارد نرم افزار MATLAB کرده و انتروپی آن را محاسبه کنیم. با دستورات ذکر شده در ادامه گزارش، فایل صوتی را وارد متلب کرده و انتروپی آن را محاسبه میکنیم و سپس میخواهیم کد کننده هافمن را به خروجی منبع اطلاعاتمان اعمال کنیم. برای انجام این کارها، پنج گام نام برده شده در صورت پروژه را طی میکنیم و در هر مرحله توضیحاتی درباره کار انجام شده در آن میدهیم.

گام اول)

در این گام به نصب نرم افزار متلبMATLAB میپردازیم. از سایت soft98.ir آن را دانلود کرده و مراحل نصب را طی میکنیم. آدرس آن را همان مقدار دیفالت در پوشه program files گذاشتم.



گام دوم)

در این مرحله، ابتدا یک فایل صوتی دلخواه از اینترنت دانلود کرده (نام موسیقی من music_bikalam_Negin هست). سپس آن را وارد نرم افزار متلب میکنیم.

```
4       [y, framePerSec] = audioread('music_bikalam_Negin.wav');
5       my_audioP = audioplayer(y, framePerSec);
6       play(my_audioP);
```

در اینجا در خط 4 با استفاده از دستورات audioread فایل صوتی را خوانده و در خط 5 و 6 با استفاده از دستورات نشان داده شده، آن را پلی میکنیم.

گام سوم)

الف) الفبای منبع همان سمبل های تولید شده در گام دوم است. این سمبل ها اعداد اعشاری هستند. در رابطه با تعداد ارقام اعشار آن، اگر آرایه X را که در متلب باز کردم، مشاهده میشود که اعداد 16بیتی و 15 رقم بعد از اعشار دارند.

فایل صوتیم دو کانال دارد که هردو رفتار مشابه ای دارن. و میتوان گفت سمبل ها یک آرایه دو بعدی با اندازه 1323000×2 هستند.

ما با یک کانال از آن کار میکنیم پس سایز سمبل ها را همان 1323000 میگیرم.

```
7       y = y(:, 1);
8       [msSize, X] = size(y);
9       fprintf('The music data size is: %d\n', msSize)
```

Name	Value	Size	Class
frameP...	44100	1x1	double
msSize	1323000	1x1	double
X	1	1x1	double
y	*1323000x1 double*	1323000x1	double

```
The music data size is: 1323000
>>
```

ب) سرعت تولید سمبل در منبع ما، همان framePerSec است که در کد مشخص شده. که همان frame per second است و آن را پرینت میکنیم.

برای به دست اوردن زمان فایل صوتی کافیست تعداد سمبل ها رو بر سرعت منبع تقسیم کنیم، حاصل را پرینت میکنیم.

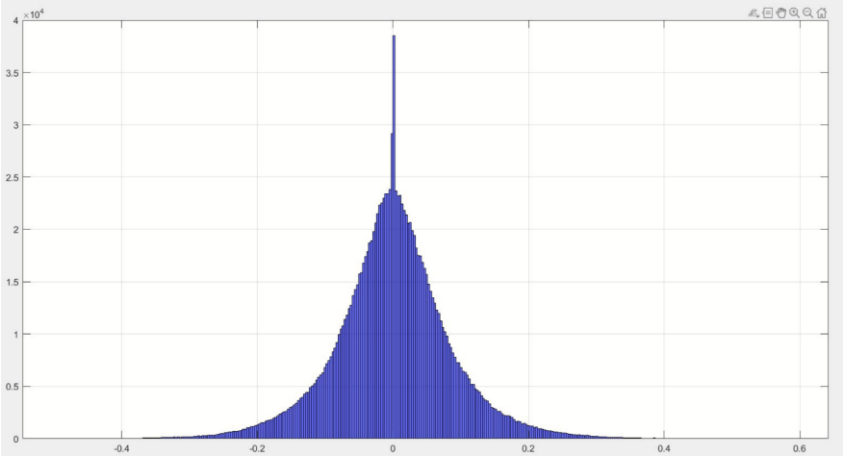
```
The music data size is: 1323000
Frame per second is: 44100
Duration is: 30.000000
>>
```

```
9   fprintf('The music data size is: %d\n', msSize)
10  fprintf('Frame per second is: %d\n', framePerSec);
11  % duration of data
12  fprintf('Duration is: %f\n', msSize/framePerSec);
```

ج) قضیه نایکوئست که در جزوه ذکر شده، برای سیگنال های صوتی هم صدق میکند و اگر با نرخ بیشتر از دو برابر بیشینه فرکانس نمونه برداری کنیم، میتوان سیگنال رو به طور کامل بازیابی کرد. حال انسان با 30 فریم بر ثانیه میتواند بدون گسستگی ببیند، برای همین فیلمی هایی که با نرخ 20 تا 30 فریم بر ثانیه پخش میشوند را بدون قطعی میبینند.

گام چهارم)

حال هیستوگرام فایل صوتی مان که همان احتمال وقوع هر سمبل است را رسم میکنیم.



میدانیم رابطه انتروپی یک منبع به صورت زیر است:

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$

پس در خط 15 احتمالها را بدست آورده و در خط 16 آنها را در رابطه فوق جایگذاری میکنیم و مجموع ضرب آنها در لگاریتم مبنای دوشان را در متغیر enthropy میریزیم.

```
15      probability = hist(y) / sum(hist(y));
16      enthropy = -sum(probability .* log2(probability));
17      fprintf("Enthropy of source is: %f\n", enthropy);
```

طبق قضیه اول شانون که اسلایدها نیز آورده شده است، میتوان N متغیر تصادفی با انتروپی H(X) را تا NH(X) بیت فشرده کرد و هیچ اطلاعاتی از دست نرود.

پس مطابق خط 18 ما میتوانیم تا msSize*enthropy/8000 بیت فشرده کنیم. یعنی فایل صوتی را میتوانیم تا حدود 290 کیلو بایت فشرده کرد.

```
17      fprintf("Enthropy of source is: %f\n", enthropy);
18      fprintf("Can compress: %f KB\n", msSize*enthropy/8000);
```

نتیجه اجرای کد به صورت زیر است:

```
The music data size is: 1323000
Frame per second is: 44100
Duration is: 30.000000
Enthropy of source is: 1.759157
Can compress: 290.920584 KB
>>
```

بنظرم نتیجه فوق معقول نیست و دلیل آن هم یکی این است که شانون احتمال ظاهر شدن سمبل ها کنار هم را مستقل از یکدیگر فرض کرده اما میتوانند مستقل از هم نبوده و به هم وابسته باشند. همچنین انتروپی از فایل نمونه برداری شده به دست امده و دقت کافی را ندارد. اینها دلایلی هستند که باعث شدند نتیجه بنظر معقول نباشد.

گام پنجم)

حال باید با روش کد گذاری هافمن، قایل صوتی را فشرده کنیم.

ابتدا مطابق خط 19 یک دیکشنری هافمن میسازیم. حال به تابع huffmanenco این دیکشنری و سیگنال اصلی را میدهیم و فایل کد گذاری شده را به ما برمیگرداند. سپس مطابق خط 23 فایل کدگذاری شده را در فایل encoded_music_Negin.wav ذخیره میکنیم.

```
19  D = huffmandict(hist(y), probability);
20  signal = randsrc(msSize, 1, [hist(y);probability]);
21  Cmp = huffmanenco(signal, D);
22  Cmp = Cmp';
23  audiowrite('encoded_music_Negin.wav', Cmp, framePerSec);
24  [x, sizeCmp] = size(Cmp);
25  fprintf("Size of signal before encoding is: %f KB.\n", msSize/8000);
26  fprintf("Size of signal after encoding is: %f KB.\n", sizeCmp/8000);
```

نتایج اجرای کد:

```
Command Window
  The music data size is: 1323000
  Frame per second is: 44100
  Duration is: 30.000000
  Enthropy of source is: 1.759157
  Can compress: 290.920584 KB
  Size of signal before encoding is: 165.375000 KB.
  Size of signal after encoding is: 299.318750 KB.
fx >>
```

طبق حاصل بدست امده از خط 26، میبینم فضای مورد نیاز برای ذخیره فایل صوتی 299 کیلو بایت است. که نزدیک به عدد بدست امده در گام چهارم (290) است و از اندازه ان قبل کدگذاری(عدد بدست امده از خط 25) بیشتر است. در یک لینک مخابراتی با سرعت 8KB/second زمان انتقال فایل به این صورت محاسبه میشود:

$$T = \frac{299}{8} = 37.375\ second$$

که یعنی 37.375 ثانیه طول میکشد.