دانشکده مهندسی کامپیوتر

هلیا وفایی – 99522347

نیم سال 1402-1401

گزارش پروژه دوم

درس انتقال داده

فهرست مطالب

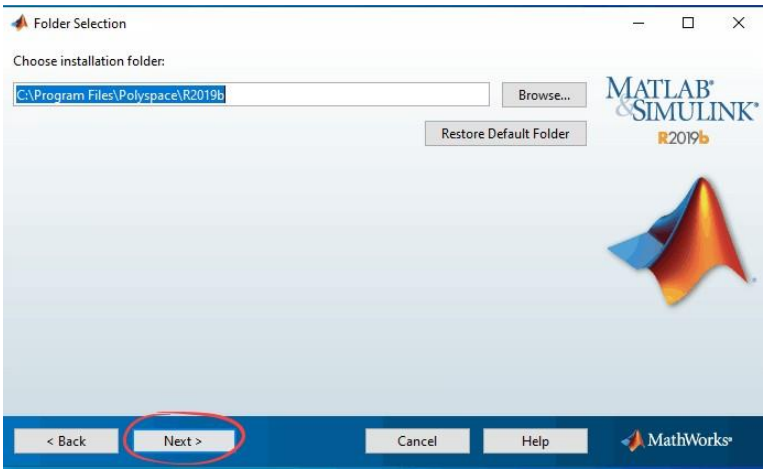
1 هدف پروژه

هدف از انجام این پروژه، بکارگیری مفاهیم آموزش‌داده‌شده در فصل نظریه اطلاعات ازجمله مفاهیم انتروپی در پردازش صوت بود. بطوری که در این پروژه باید ابتدا با ابزار وارد کردن و خواندن یک فایل صوتی در نرم‌افزار MATLAB آشنا می‌شدیم. سپس با استفاده از ابزارها و دستورهای مناسب، انتروپی فایل صوتی مربوطه را محاسبه کنیم و در مرحله آخر نیز با استفاده از یکی از کدکننده‌های بدون اتلاف کدکننده‌ی هافمن) آن را به خروجی فایل صوتی اعمال کنیم.

این پروژه درعمل شامل ۵ گام مختلف بود که در هر گام به پیاده‌سازی موارد خواسته‌شده پرداخته شد که در این گزارش به تفصیل کارهای انجام‌شده در هر گام خواهیم پرداخت.

2 گام اول

در گام اول، فارغ از نوع سیستم عاملی که در اختیاز داریم، باید به نصب نرم‌افزار MATLAB می‌پرداختیم.



شکل 1: نصب نرم‌افزار

3 گام دوم

در این گام، یک فایل صوتی با پسوند Wav را از حافظه کامپیوتر می خوانیم و در نرم‌افزار متلب وارد می کنیم. این کار را با استفاده از دستور audioread(path) انجام می دهیم.

در تصویر زیر، کدهای مربوط به این قسمت را مشاهده می‌کنید. همانطور که مشخص است، تابع audioread دو خروجی دارد که درواقع خروجی اول آن (y)، دیتای نمونه برداری شده از فایل صوتی و خروجی دوم آن (Fs) مقدار rate برحسب فریم بر ثانیه برای داده‌ی نمونه برداری شده است.

```
% read audio from storage
[y, Fs] = audioread('tanhatarin_ashegh.wav');

% create an audioplayer object from our audio and play it
player = audioplayer(y, Fs);
play(player);
```

شکل 2: خواندن فایل صوتی از حافظه کامپیوتر و اجرای آن

4 گام سوم

در پاسخ به سوالات مربوط به این گام می‌توانیم بگوییم:

الف)

الفبای منبع در این حالت (که درواقع همان سمبل‌های تولید شده در قسمت قبل هستند)، بصورت عدد و آن هم از جنس اعشاری هستند. اگر آرایه‌ی y را در متلب بازکنیم، می بینیم که اعداد مربوطه همگی با دقت ۵۱ رقم اعشار هستند و این به این معنی است که سمبل‌های تولید شده ۶۱ بیتی هستند.

این سمبل‌ها درواقع یک آرایه دوبعدی با اندازه‌ی 2 × 1323000می باشند. درواقع فایل صوتی ما دارای دو کانال است که هر دو کانال تقریبا رفتار برابری دارند. (با رسم هیستوگرام آنها متوجه می شویم) پس برای ادامه کار، فقط با کانال اول کار می‌کنیم و درواقع اندازه سمبل‌ها را 1323000 درنظر می‌گیریم.

ب)

سرعت تولید سمبل در منبع اطلاعاتی ما (فایل صوتی) مقدار 00144 فریم بر ثانیه است. حال با داشتن مقدار سرعت منبع و تعداد سمبل‌ها، اگر تعداد سمبل‌ها را بر سرعت منبع تقسیم کنیم، مشاهده می‌کنیم که مقدار آن برابر با مقدار زمان فایل صوتی ما خواهد شد.

```
% print size of data (symbols) of our audio file
[data_size, x] = size(y);
fprintf('Size of data = %d\n', data_size)

% print the value of FPS of our audio file
fprintf('Fps = %d\n', Fs);

% Calculate the duration of our audio file
fprintf('Audio duration = %f\n', data_size / Fs);
```

شکل 3: دستورات مربوط به محاسبه کردن زمان فایل صوتی

```
Size of data = 1323000
Fps = 44100
Audio duration = 30.000000
```

شکل 4: نتیجه ی اجرای دستورات

ج)

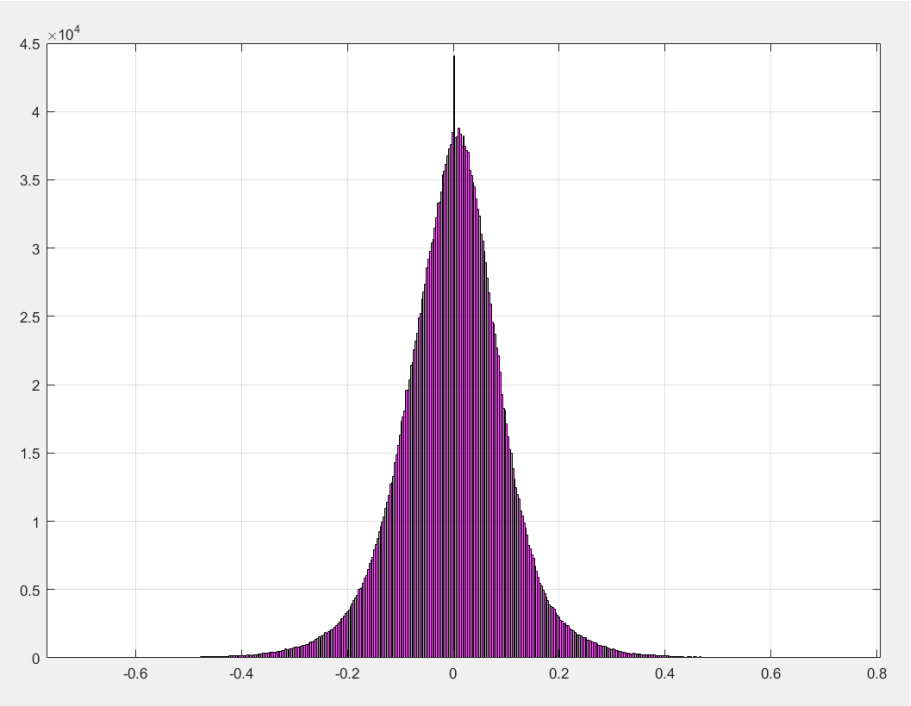
برای بیان بهتر دلیل این موضوع بهتر است گذری به قضیه نایکوئست داشته باشیم. همانطور که می‌دانیم، اگر نرخ نمونه برداری بزرگتر از دو برابر مقدار فرکانس بیشینه باشد، در سمت گیرنده می‌توان سیگنال را بطور کامل بازیابی کرد و به سیگنال اصلی رسید.

این موضوع در سیگنال‌های صوتی نیز برقرار است. در فیلم‌هایی که با 02 تا 03 فریم بر ثانیه پخش می‌شوند نیز، با توجه به اینکه مغز انسان تا مقدار 03 فریم بر ثانیه را می‌تواند بدون گسستگی ببیند، این اتفاق نخواهد افتاد و به همین دلیل ما فیلم ها را هم بدون هیچ گونه قطعی می‌بینیم.

5 گام چهارم

در این گام، به رسم هیستوگرام فایل صوتی مربوطه می‌پردازیم. برای رسم هیستوگرام که درواقع نشان‌دهنده ی احتمالات وقوع هر سمبل در فایل صوتی است، از دستور histogram(y, 'FaceColor', color) استفاده کرده‌ام.

در شکل زیر هیستوگرام رسم‌شده را می‌بینید:



شکل 5: هیستوگرام فایل صوتی

برای به‌دست آوردن مقدار انتروپی منبع، همانطور که می‌دانیم مقدار انتروپی از فرمول زیر به دست می‌آید:

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

پس کاری که باید انجام دهیم، این است که آرایه ی احتمالات را به دست آوریم و مجموع حاصل ضرب احتمالات در log_2 آن را به دست آوریم و آن را قرینه کنیم که این کار را در قطعه کد زیر انجام داده‌ایم.

```
36      % find the probability of our data
37 -    prob = hist(y) / sum(hist(y));
38
39      % calculate the entropy value from the probability
40 -    enthropy = -sum(prob .* log2(prob));
41 -    fprintf("Enthropy = %f\n", enthropy);
42
```

```
Command Window
  Size of data = 1323000
  Fps = 44100
  Audio duration = 30.000000
  Enthropy = 1.686061
```

شکل 6: دستورات مربوط به محاسبه‌ی انتروپی

همانطور که مشاهده می‌شود، مقدار انتروپی برابر با 1.686061 به دست آمده است.

قضیه اول شانون یا قضیه کدگذاری کانال به ما می‌گوید N متغیر تصادفی با انتروپی H(X) را می‌توان حداکثر تا NH(X) بیت فشرده‌سازی کرد و اگر بیشتر از این مقدار فشرده کنیم، ممکن است موجب از بین رفتن داده‌ها شود. اگر حاصل‌ضرب انتروپی در تعداد سمبل‌ها را پیدا کنیم، این مقدار برابر با 278.83KB = 1323000/8000 × 1.686061 خواهد بود. یعنی شانون می‌گوید می‌توانیم این فایل صوتی را تا مقدار 872 کیلوبایت فشرده کنیم.

نتیجه بدست‌آمده منطقی نیست. دلیل آن هم می‌تواند این باشد که انتروپی بدست آمده از فایل نمونه برداری شده به دست آمده است و ممکن است دقت کافی را نداشته باشد. هم چنین چون نمونه برداری کرده‌ایم، مقادیر سطوح انرژی می‌توانند بسیار نزدیک به همدیگر باشند که این تفاوت نادیده گرفته می‌شود. همچنین ممکن است احتمال ظاهرشدن سمبل‌های منبع اطلاعاتی ما در کنار همدیگر مستقل از هم نباشند و به همدیگروابسته باشند؛ درصورتی که شانون آنها را مستقل از همدیگر فرض کرده است. بنابراین این مرز دقیق و مناسب نمی‌باشد.

6 گام پنجم

در این گام، اقدام به فشرده‌سازی فایل صوتی با استفاده از روش گدگذاری هافمن می‌کنیم.

برای این کار، ابتدا با استفاده از تابع huffmandict(hist, prob) یک دیکشنری هافمن را بر اساس سمبل‌ها و اطلاعات آن‌ها می‌سازیم و سپس این دیکشنری را به همراه سیگنال اصلی به تابع huffmanenco می‌دهیم تا فایل کدگذاری شده را به ما برگرداند و درنهایت هم فایل کدگذاری شده

را ذخیره می‌کنیم.

```
36      % find the probability of our data
37 -    prob = hist(y) / sum(hist(y));
38
39      % calculate the entropy value from the probability
40 -    enthropy = -sum(prob .* log2(prob));
41 -    fprintf("Enthropy = %f\n", enthropy);
42
```

Command Window

```
Size of data = 1323000
Fps = 44100
Audio duration = 30.000000
Enthropy = 1.686061
```

شکل 7: دستورات مربوط به کدگذاری هافمن و نتایج مربوط به آن

همانطور که مشاهده می‌شود، فضای مورد نیاز برای ذخیره‌ی فایل صوتی مقدار 289 KB به دست آمده است که به عددی که در گام قبل توسط قضیه اول شانون به دست آمد (278 KB) خیلی نزدیک است. اما اگر اندازه‌ی آن را با اندازه‌ی سیگنال قبل از کدگذاری (165 KB) مقایسه کنیم، متوجه می شویم که اندازه‌ی آن افزایش یافته است. برای محاسبه ی زمان لازم برای انتقال این فایل در یک لینک مخابراتی با سرعت 64 Kbit/s یا 8 KBytes/s کافی است مقدار به دست‌آمده برای اندازه‌ی سیگنال را بر این مقدار تقسیم کنیم:

$$T = \frac{289KBytess}{8KBytes} = 36.125s$$