UNIVERSIDADE DE COIMBRA

Teoria da informação

Trabalho Pratico nº1

Entropia, Redundância e Informação Mútua

André Monteiro, 2013171072

Ana Fidalgo, 2013132819

Ricardo Costa, 2013154238

Introdução

A Teoria da informação é um ramo da teoria da probabilidade e da matemática estatística que lida com sistemas de comunicação, transmissão de dados, criptografia, codificação, teoria do ruído, correção de erros, compressão de dados, etc.
Neste trabalho, mais especificamente, foram resolvidos exercícios em MATLAB com os conhecimentos adquiridos na área da entropia, a redundância de uma fonte e a informação mútua.

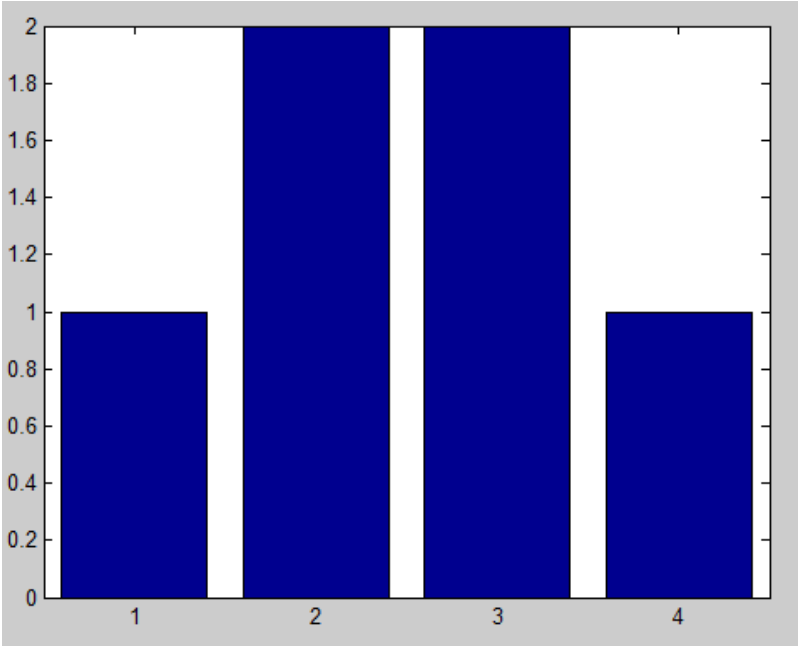
Questão 1

No exercício 1 era-nos pedido que escrevêssemos uma rotina para determinar o histograma de ocorrência dos símbolos de uma dada fonte de informação P e um alfabeto A = {$a_1 \ldots a_n$}.

Criamos uma função histograma, usando como argumentos o alfa (o alfabeto) e a fonte, de seguida criamos um array (h) com o comprimento do alfabeto e criamos um ciclo for de modo a devolver o outro array com as posições de uma determinado símbolo aparece no array fonte e h(k) toma o valor do array j, ou seja o numero de ocorrências de alfa(k) na fonte. Por fim usamos a função bar() para criar o histograma de h. Imaginemos que corremos a função histograma da seguinte forma:

histograma([1,2,3,2,3,4], [1,2,3,4])

Tendo como resposta o número de ocorrências (1,2,2,1), obteríamos o seguinte histograma:



Questão 2

Neste exercício foi-nos pedido para escrever um código que dada uma fonte de informação P com um alfabeto A = $\{a_1 \ldots a_n\}$ que determinasse o limite mínimo teórico para o numero medio de bits por símbolo, ou seja a entropia.

Criamos uma função entropy com os argumentos alfa (alfabeto) e fonte e chamamos a função histograma criada no exercício anterior. De seguida criou-se uma condição para eliminar os zeros pois o logaritmo de zero de base dois é inexistente e calculou-se a probabilidade de cada valor aparecer a partir dos dados anteriormente referidos.

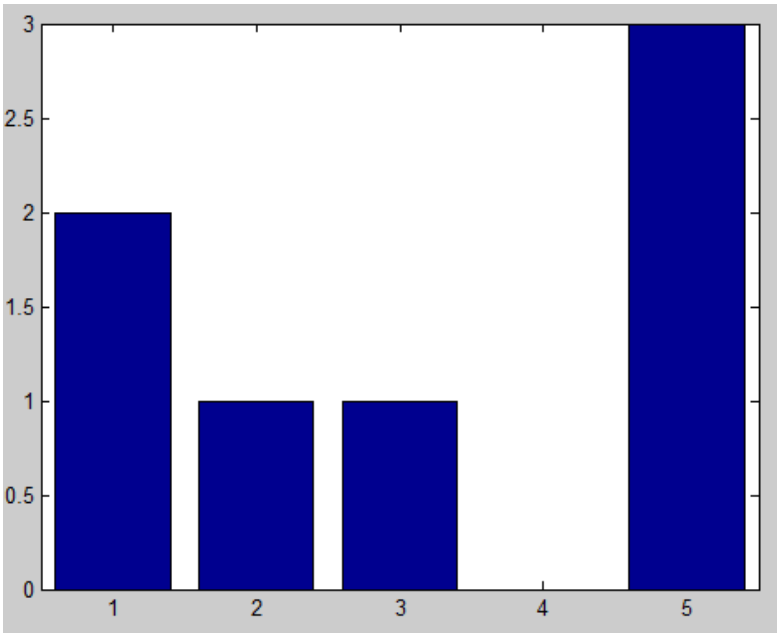
Finalmente aplicou-se a fórmula da entropia:

$$-\sum_{i=1}^{n} P(a_i) \log_2 P(a_i)$$

Imaginemos que corremos a função da seguinte forma:

entropy([1,1,2,3,5,5,5], [1,2,3,4,5])

Obtemos o seguinte histograma:



E a seguinte entropia: 1.8424

Questão 3

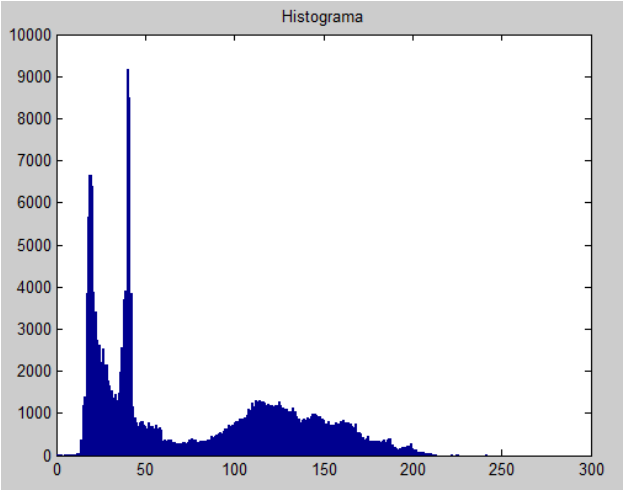
No exercício 3 é-nos pedido para usar as rotinas desenvolvidas nas alíneas anteriores para determinar a distribuição estatística das fontes que nos foram fornecidas.

Sendo assim foram criadas 3 rotinas em MATLAB, uma para os ficheiros de som, outra para os de texto e ainda outra para ficheiros de imagem.

Na rotina em que se determinou a distribuição estatística de texto procedeu-se da seguinte forma: Começou-se por abrir o ficheiro de texto e guardar na fonte todos os caracteres que o compunham excetuando os espaços em branco. Criou-se um alfabeto alfa de 'a' a 'z', minúsculas e maiúsculas, e finalmente chamou-se a função obtendo-se assim o resultado esperado.

Seguidamente determinamos a distribuição estatística de uma fonte de imagem. Primeiro começamos por calcular a intensidade de cada pixel usando a função imread() e devolvemos o resultado como argumento para um array img[ ], a seguir, criamos um alfabeto de "0 a 255" porque existem entre 0 a 255 tons de cinza ou seja 256 símbolos possíveis e por fim calculamos a entropia usando a função entropia que usamos no exercício 2 e que também já nos fornecia o histograma.

Por exemplo, no ficheiro kid.bmp obtemos o histograma:



E uma entropia de 6.954143.

Por fim, determinamos a distribuição estatística de uma fonte de som. Primeiro, usando a função "wavread()" começamos por determinar a amplitude do aúdio e guardamos em "Y", determinamos ainda o número de bits por amostra e guardamos em "nbits" . De seguida, precisamos descobrir a diferença entre cada valor do alfabeto, por isso, usamos a seguinte função: 2/(2^nbits) , onde o 2 equivale ao espaço total do alfabeto e o 2^nbits equivale ao tamanho de cada espaço. Por fim conseguimos determinar o alfabeto, que vai ser entre -1 e 1-h com intervalo de h entre cada nível e no fim chamamos a função entropia realizada anteriormente e calculamos a entropia da fonte de som com o alfabeto antes referido.

Obtivemos então os seguintes resultados neste exercício:

FONTE	Entropia (bits/simbolo)	Entropia Maxima	Taxa de Compressão
'Kid.bmp'	6.954143	8	13,07%
'Homer.bmp'	3.465865	8	56.68%
'HomerBin.bmp'	0.644781	8	91.94%
'GuitarSolo.wav'	7.35802	8	8,02%
'English.txt'	4.194289	8	47,57%

Na Imagem 2, concluímos que existe uma grande abrangência de tons entre o preto e o branco na fonte analisada, logo um grande número de bits a representarem esta gama de cores.

Na Imagem 3, concluímos que existe uma grande percentagem de bits a representar o preto, uma minoritária percentagem de tons cinzentos e uma ainda menor percentagem de tons perto do branco.

Na Imagem 4, concluímos que apenas existem duas cores, o preto e o branco. Não é, portanto, de admirar que tenhamos conseguido uma taxa de compressão tão grande: só precisamos de um bit para representar duas cores.

Na Imagem 5 observamos que a distribuição probabilística apresenta uma concentração no centro do gráfico. Isto justifica-se pela predominância de valores próximos de zero nos períodos de silêncio e baixa amplitude.

Por fim, na Imagem 6, concluímos que existem caracteres que aparecem com maior frequência e que há uma larga quantidade de bits que pode ser poupada.

Logo concluímos que os gráficos com maior dispersão são os que apresentam maior entropia. Concluímos então que é possível comprimir de forma não destrutiva com o número de bits médio teórico dado pelos valores da entropia.

Questão 4

No exercício 4 determinamos o número médio de bits por símbolo para cada uma das fontes de informação antes referenciadas, mas usando rotinas de codificação de Huffman.

Na fonte de informação de texto, primeiro começamos por criar uma unção Huffman com fonte e alfa como argumentos e criamos um alfabeto próprio para este tipo de fonte, ou seja, neste caso usamos um alfabeto de "a" a "z", com letras minúsculas e maiúsculas. De seguida usando a função histograma(), criamos um array onde vão estar inseridas as ocorrências da fonte no alfabeto e usando a função hufflen() que estava incluída no enunciado, criamos uma array "bits" que contém o número de bits necessários para a codificação de Huffman e por fim calculamos o número médio de bits por símbolo simplesmente multiplicando o número de bis necessários à codificação pela probabilidade de ocorrência de símbolos.

Na fonte de informação de imagem, usamos a função Huffman usada anteriormente e criamos um alfabeto próprio para este tipo de fonte, ou seja, neste caso usamos um alfabeto de "0" a "255", porque existem 255 tons de cinza ou seja 256 símbolos possíveis. De seguida usando a função histograma(), criamos um array onde vão estar inseridas as ocorrências da fonte no alfabeto anteriormente referido e usando a função hufflen() que estava incluída no enunciado, criamos uma array "bits" que contém o número de bits necessários para a codificação de Huffman e por fim calculamos o número médio de bits por símbolo da mesma que calculamos para uma fonte de texto.

Na fonte de informação de som, voltamos a usar a função Huffman e criamos um alfabeto próprio para este tipo de fonte, ou seja, usamos um alfabeto de "-1" a "1-h", com intervalo de h entre cada nível . De seguida usando a função histograma(), criamos um array onde vai estar inserido as ocorrências da fonte no alfabeto antes referido e usando a função hufflen() que estava incluída no enunciado, criamos uma array "bits" que contém o número de bits necessários á codificação de Huffman e por fim calculamos o número médio de bits por símbolo da mesma maneira que utilizamos anteriormente.

FONTE	Entropia (bits/simbolo)	Huffman (bits/simbolo)
'Kid.bmp'	6.954143	6.983223
'Homer.bmp'	3.465865	3.548316
'HomerBin.bmp'	0.644781	1.000000
'GuitarSolo.wav'	7.35802	7.379079
'English.txt'	4.194289	4.200579

Concluímos que, para obter um código de Huffman de variância mínima, teríamos de colocar os símbolos combinados na lista usando a ordem mais elevada possível, pois a variância é mínima num código de Huffman quando, na formação da árvore de Huffman, damos preferência aos símbolos que têm menor comprimento (por resultarem, ou não, da junção de vários símbolos).

Questão 5

Neste exercícios foi-nos pedido para calcular exatamente a mesma coisa que no exercício 3 mas agrupando os símbolos da fonte dois a dois. Se tivermos uma matriz temos de a transformar num vetor usando a função reshape() e calculamos o comprimento da fonte. Depois criamos uma matriz quadrada em que cada posição corresponde a um par de símbolos, em seguida preenchemos a matriz com as ocorrências desses pares na fonte e depois com o número de ocorrências calculamos a probabilidade de cada um desses pares, depois de eliminar os zeros da matriz podemos assim calcular a entropia e usamos a função bar() para criar o diagrama de ocorrências.

Analisando os resultados e comparando com a primeira entropia calculada, observamos uma melhor taxe de compressão e quanto à entropia, que em alguns casos diminui quase para metade. Confirma-se que, neste caso, o agrupamento de símbolos é muito benéfico, pois permitiu diminuir a entropia.

FONTE	ENTROPIA (bits/símbolo) – Considerando agrupamento de 2 símbolos	Taxa de compressão
'Kid.bmp'	4.920088	38,50%
'Homer.bmp'	2.419832	69,75%
'HomerBin.bmp'	0.390731	95,12%
'GuitarSolo.wav'	5.780817	27,74%
'English.txt'	3.571288	55,36%

Questão 6

a) O objetivo deste exercício é comparar uma fonte pequena (query) com a outra fonte (target).

Começamos por determinar o comprimento do query, da fonte e do alfabeto. Seguidamente calculamos o número de vezes que comparamos a fonte com o query em diferentes janelas. Calculamos a entropilhada query e da janela atual. Em seguida criamos uma matriz com o comprimento do alfabeto e preenchemo-la com o número de casos de cada combinação ocorrer.

Transformamos essa matriz num vetor e calculamos a entropia desse vetor. Utilizando os valores anteriores aplicamos a fórmula da informação conjunta.

b) Pretende-se que utilizemos a função criada no exercício anterior para comparar os sons "guitarSolo.wav" (como query) aos sons "target01 – repeat.wav" e "target02 – repeatNoise.wav", que funcionaram como target. Existem 3 picos de informação mútua, esses picos são maiores no caso do primeiro som, o "target01", que regista 'picos' de 7.3207, 7.2789 e 7.2806 contra os valores 2.7113, 2.6934 e 2.6946 do "target02". Podemos então concluir que o primeiro ficheiro se assemelha mais ao ficheiro "guitarSolo.wav".

c) Neste exercício, pretendia-se estudar a evolução da informação mútua ao longo de dois ficheiros áudio: o ficheiro "guitarSolo.wav", como query, e cada um dos ficheiros "Song0*.wav", como target. Pretendia-se ainda saber o máximo valor para a informação mútua em cada uma destas análises.

Valores máximos de Informação Mútua:

1. Song06 - 7.33838 bits
2. Song07 - 6.31305 bits
3. Song05 - 3.96175 bits
4. Song04 - 0.40970 bits
5. Song02 - 0.37768 bits
6. Song03 - 0.30445 bits
7. Song01 - 0.25784 bits

Daqui se conclui então que a música que mais se assemelha ao ficheiro "guitarSolo.wav" é a Song06, com um valor bem mais elevados que as músicas das posições mais baixas da tabela.

Anexo de Resultados

Ficheiro: kid.bmp
Entropia = 6.954143e+00
Huffman = 6.983223e+00
Entropia com combinações = 4.939796

Ficheiro: homer.bmp
Entropia = 3.465865e+00
Huffman = 3.548316e+00
Entropia com combinações = 2.507404e+00

Ficheiro: homerbin.bmp
Entropia = 6.447814e-01
Huffman = 1
Entropia com combinações = 3.878529e-01

Ficheiro: guitarSolo.wav
Entropia = 7.358020e+00
Huffman = 7.379079e+00
Entropia com combinações = 5.806828e+00

Ficheiro: english.txt
Entropia = 4.279827e+00
Huffman = 4.303279e+00

Informação mútua

guitarSolo.wav vs target01 - repeat.wav

7.3207 0.3289 0.3354 0.3297 7.2789 0.3289 0.3355 0.3293 7.2806

guitarSolo.wav vs target02 - repeatNoise.wav

2.7113 0.3354 0.3379 0.3366 2.6934 0.3363 0.3388 0.3340 2.6946

Simulador

Song01
0.2552 0.2578
Máximo:
0.2578

Song02
0.3777

Máximo:
0.3777

Song03
0.3045 0.3022
Máximo:
0.3045

Song04
0.4097 0.4067
Máximo:
0.4097

Song05
Columns 1 through 11

3.9618 0.3278 0.3282 0.3290 0.3432 0.3359 0.3303 0.3372 0.3348
0.3345 0.3337

Columns 12 through 22

0.3407 0.3429 0.3337 0.3442 0.3292 0.3322 0.3280 0.3355 0.3350
0.3421 0.3320

Columns 23 through 29

0.3407 0.3525 0.3344 0.3335 0.3459 0.3296 0.3220

Máximo:
3.9618

Song06

Columns 1 through 11

7.3384 0.3264 0.3301 0.3281 0.3437 0.3345 0.3307 0.3391 0.3308
0.3350 0.3361

Columns 12 through 22

0.3379 0.3397 0.3370 0.3440 0.3318 0.3294 0.3251 0.3348 0.3353
0.3413 0.3338

Columns 23 through 29

0.3425 0.3527 0.3328 0.3337 0.3434 0.3315 0.3234

Máximo:
7.3384

Song07
Columns 1 through 11

6.3131 0.1778 0.1818 0.1795 0.1920 0.1855 0.1797 0.1937 0.1848
0.1856 0.1843

Columns 12 through 22

0.1879 0.1877 0.1852 0.1971 0.1854 0.1836 0.1832 0.1884 0.1824
0.1895 0.1839

Columns 23 through 29

0.1936 0.2100 0.1877 0.1844 0.1940 0.1832 0.1833

Máximo:
6.3131

Resultados da simulação:

7.3384

6

6.3131

7

3.9618

5

0.4097

4

0.3777

2

0.3045

3

0.2578

1