Relatório

Trabalho Prático nº1

Teoria da Informação

Licenciatura em Engenharia Informática

Entropia, Redundância, Informação Mútua

João Moreira – 2015230374

Pedro Gonçalves – 2013150557

Ricardo Tavares – 2014230130

Índice

Introdução

Neste primeiro trabalho prático da disciplina de Teoria da Informação, pretendemos explorar melhor o conceito de entropia, redundância e informação mútua.

Nas páginas que se seguem, explicamos de forma geral todos estes conceitos, e conseguimos visualizar na prática os conceitos expostos, como por exemplo, todos estes valores iram variar quando estudamos ficheiros com semelhanças ou ficheiros que não partilhem qualquer tipo de semelhança.

Exercício 1

Primeiramente, o exercício pede uma função que receba como parâmetros de entrada duas matrizes (uma fonte de informação "P" e um alfabeto "A"). Esta função devolve os dados necessários para fazer um histograma[1].

Para tal, criámos uma função (histograma.m) que recebe esses parâmetros como argumentos. Para a fonte de informação transformamo-la numa matriz unidimensional e passamos o alfabeto a double. De seguida usamos a função do MatLab "histc" que calcula o número de ocorrências de cada símbolo do alfabeto na respetiva fonte, devolvendo numa matriz essas ocorrências. Para representar o histograma usamos a função "bar" sobre a matriz obtida com o respetivo alfabeto.

[1] - *Um histograma é a representação gráfica, em colunas, de um conjunto de dados consoante a frequência no intervalo da classe.*

Exercício 2

Neste exercício, queremos calcular a entropia[2].
A fórmula que estamos a usar é:

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

No ficheiro entropia.m, começamos por obter a matriz de ocorrência que vem da função que foi feita no exercício 1 e recriamos uma matriz de zeros onde teremos as probabilidades de cada símbolo do alfabeto na fonte de informação.

Como necessitamos do tamanho da fonte, fazemos a multiplicação das linhas pelas colunas.

Após percorrermos a matriz de ocorrência, colocamos em cada índice da mesma, a probabilidade de ocorrer cada um dos símbolos, sendo que as probabilidades que são maiores que zero são colocadas em x1.

Por fim utilizamos a fórmula da entropia e enviamos o valor da entropia através da variável de retorno.

[2] – *A entropia é o limite mínimo teórico para o número médio de bits por símbolo.*

Exercício 3

Para resolver este exercício começámos por ler as fontes:

Imagens (imread):

- Lena.bmp
- CT1.bmp
- Binaria.bmp

Áudio (audioread e audioinfo):

- saxriff.wav

Texto (fopen e fscanf):

- Texto.txt

Depois definimos os alfabetos:

Imagens de 0 a 255, com passo 1

Áudio de -1 a (1-d), compasso d, sendo d=1/(2^nbits)

Texto de 'A' a 'Z' em maiúsculas e minúsculas (['A':'Z' 'a':'z'])

Depois, fazemos o histograma e entropia individualmente para cada um dos ficheiros:

Imagens:

Estas têm uma fonte de informação que se traduz na codificação da cor dos pixéis em 8 bits, sendo que o seu alfabeto é composto por 256 símbolos logo está contido no intervalo de [0, 255].

- Lena.bmp

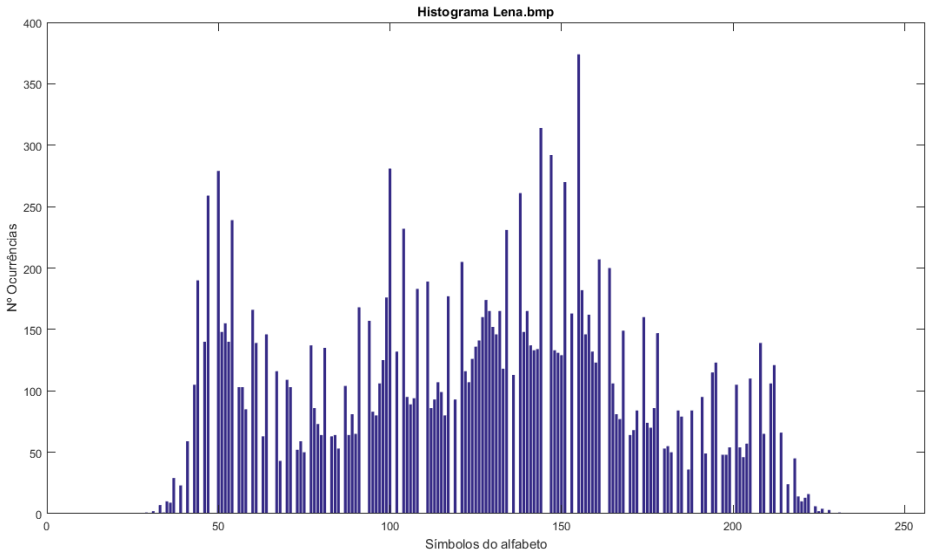

Figura 1 – Histograma de ocorrência de Lena.bmp

Em média, é necessário 6,9153 bits p/símbolo para codificar esta fonte. Este é um valor que se encontra relativamente perto do valor máximo teórico de 8 bits p/símbolo.

Neste histograma conseguimos observar uma dispersa distribuição sobre a incidência de cada símbolo visto que a imagem é composta por, maioritariamente pixéis em tons de cinzento, logo concluímos que isto implica uma incerteza elevada.

- CT1.bmp

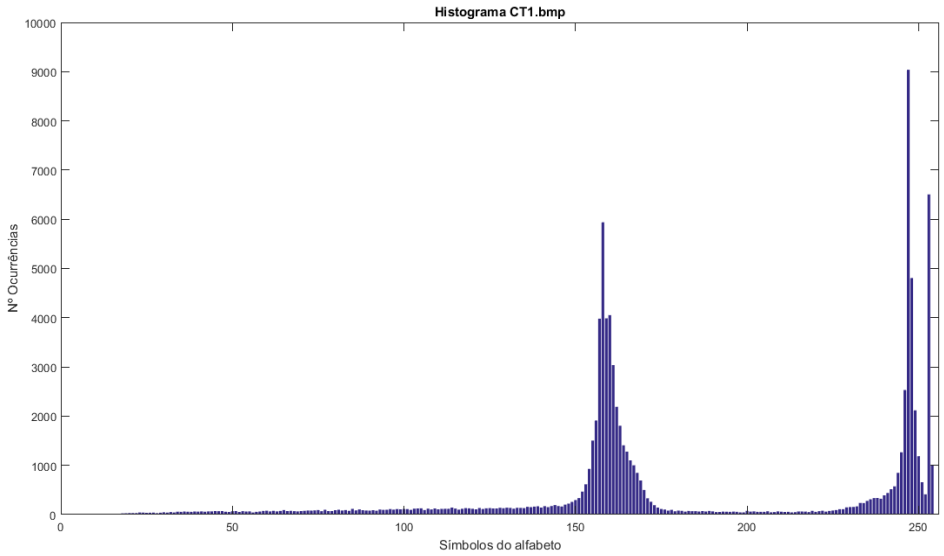

Figura 2 – Histograma de ocorrência de CT1.bmp

Sendo que na imagem anterior tínhamos variados tons de cinza, nesta temos uma variedade menor, sendo necessário 5.9722 bits p/símbolo para codificar esta fonte, sendo este um valor mais afastado do máximo de 8 bits p/símbolo.

O histograma apresenta uma dispersão menor, sendo que apenas têm picos na zona do branco e cinza.

- Binaria.bmp

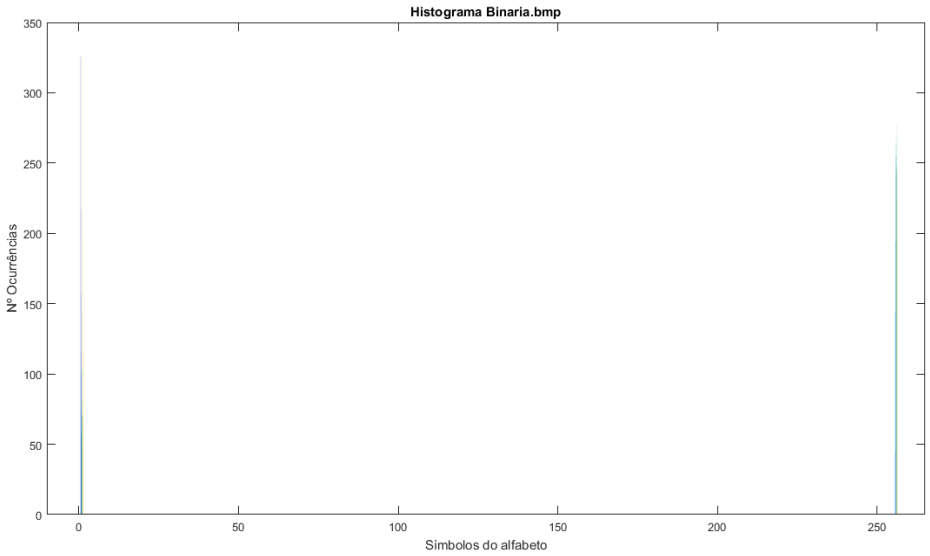

Figura 3 – Histograma de ocorrência de Binaria.bmp

Neste caso, apenas necessitamos de 0.9755 bits p/símbolo para codificar a fonte. O afastamento dos 8 bits p/símbolo é assim tão grande pelo simples facto de termos apenas 2 cores: o preto e o branco, ou seja, ou é 0 ou 255. O histograma mostra-nos exatamente isso visto que nos apresenta esses dois símbolos apenas, sendo que isto implica uma incerteza baixíssima.

Áudio:

O áudio é uma fonte de informação com as suas amostras codificadas em 8 bits, sendo que o seu alfabeto varia entre [-1, 1] com intervalos de d (2/(2^nbits)).

- saxriff.wav

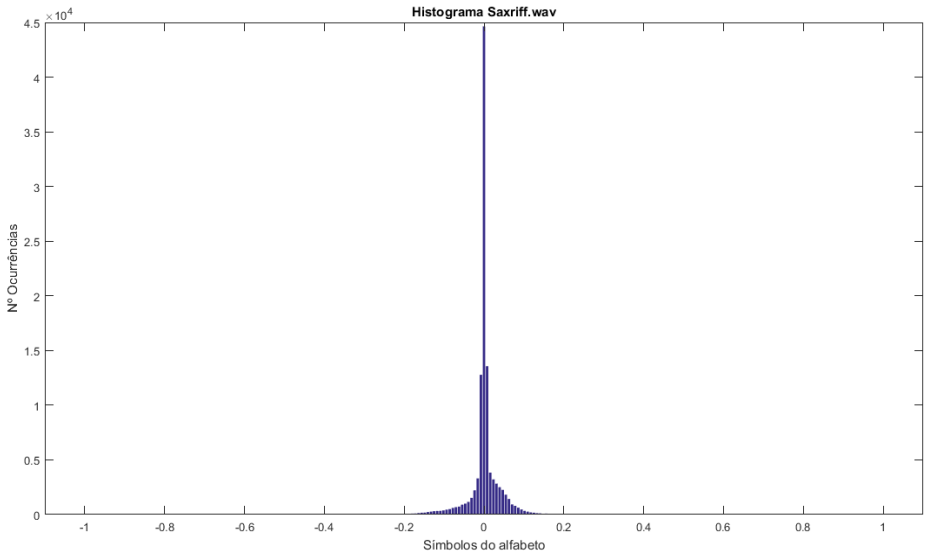

Figura 4 – Histograma de ocorrência de saxriff.wav

Neste caso necessitamos de 3.5356 bits p/símbolo para codificar a fonte.

Este histograma tem uma distribuição pouco dispersa, sendo que incide maioritariamente no 0. Isto deve-se ao facto do ficheiro ser quase monocórdico.

Texto:

Esta fonte de informação é um Texto em português com um alfabeto composto por 52 símbolos, símbolos estes compreendidos entre [A, Z] e [a, z], sendo que estes símbolos estão a ser trabalhados nos seus correspondentes códigos ASCII.

- Texto.txt

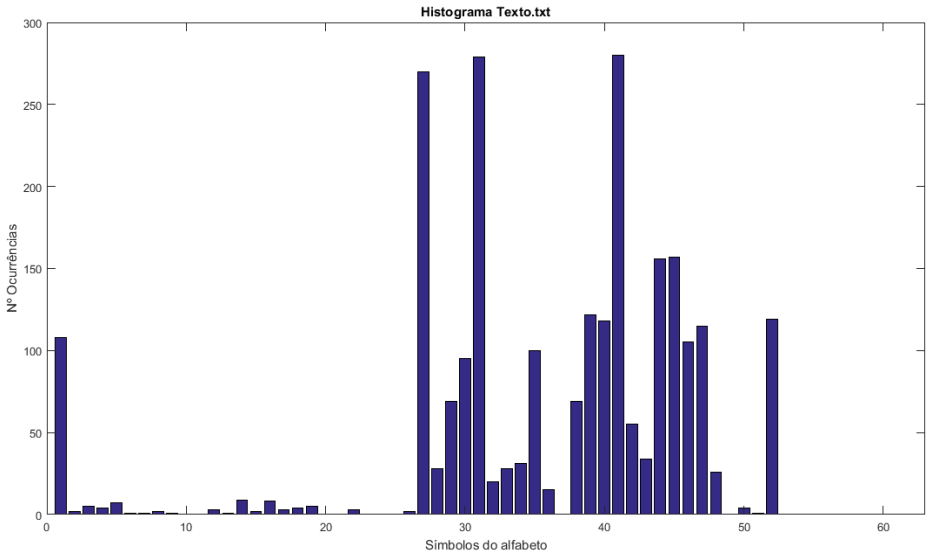

Figura 5 – Histograma de ocorrência de Texto.txt

Para esta fonte de informação, o resultado obtido foi diferente do que era suposto, sendo que neste caso necessitamos de 4.3130 bits p/símbolo para codificar a fonte. A diferença altera todos os resultados em que o ficheiro Texto.txt está presente como ficheiro a ser estudado, esta diferença pode vir da compressão utilizada para extrair o ficheiro, sendo que não podemos afirmar com toda a certeza.

Este histograma tem uma distribuição bastante dispersa, sendo que incide maioritariamente nas letras mais utilizadas na língua portuguesa.

Pergunta:

🕮 Será possível comprimir cada uma das fontes de forma não destrutiva? Se Sim, qual a compressão máxima que se consegue alcançar? Justifique.

R: Teoricamente, estas fontes são possíveis de comprimir de forma não destrutiva, visto que todas elas, são nos dadas de forma não comprimida e cada símbolo é codificado com um determinado número de bits.

Neste caso, cada uma das fontes tem um valor mínimo teórico para o número médio de bits para o qual pode codificar um símbolo.

Se existir um algoritmo capaz de codificar as fontes para estes limites que encontramos, então obteríamos as seguintes taxas de compressão:

Fonte	Bits	Entropia	Entropia Arredondada	Taxa de Compressão
Lena.bmp	8	6.9153	7	12.5%
CT1.bmp	8	5.9722	6	25%
Binaria.bmp	8	0.5755	1	87.5%
Safriff.wav	8	3.5356	4	50%
Texto.txt	6	4.3130	4	33.3%

Tabela 1 – Fonte, Bits de codificação, Entropia e Entropia Arredondada e Taxa de Compressão

O problema disto é no mundo real não conseguirmos este tipo de valores, visto que estamos limitados pelos códigos ótimos que satisfazem esta condição:

$$H(S) \leq L < H(S) + 1$$

Isto significa que podemos não conseguir comprimir de forma não destrutiva certos ficheiros que se encontrem muito perto do valor limite.

Exercício 4

Neste exercício utilizamos um ficheiro com o nome de hufflen.m. Este ficheiro contem a codificação de Huffman, sendo este um código de compressão que utiliza as probabilidades para atribuir um número de bits a cada elemento do alfabeto. Portanto: os que têm maior probabilidade de ocorrência são codificados com um menor número de bits e os que têm menor probabilidade de ocorrência, um maior número de bits.

Os resultados obtidos foram os seguintes:

Fonte	Entropia	Huffman - Média de Bits	Huffman - Variância
Lena.bmp	6.9153	6.9425	0.0272
CT1.bmp	5.9722	6.0075	0.0353
Binária.bmp	0.5755	1	0.4245
Saxriff.wax	3.5356	3.5899	0.0543
Texto.txt	4.3130	4.3466	0.0336

Tabela 2 – Resultados do Código de Huffman

Os valores são todos relativamente próximos aos do exercício 3, com exceção do ficheiro Binaria.bmp. Este apresenta um valor significativamente maior, isto porque o ficheiro tem apenas dois valores observáveis no alfabeto de 0 a 255, sendo estes valores o 0 e o 255. Com huffman, a probabilidade de ocorrer os valores entre 1 a 254 é nula, portanto não lhes é concedida uma representação, sendo que o número de bits utilizados para os representar é 0, logo a variância do número de bits é bem mais elevada.

Pergunta:

📖 Será possível reduzir-se a variância? Se sim, como pode ser feito em que circunstância será útil?

R: Sim, é possível reduzir-se a variância.

Para fazermos isto temos de equilibrar a árvore de Huffman, sendo que quando existem mais de dois nós com probabilidades iguais, devemos escolher os que estão mais a baixo e mais alto na árvore e combiná-los. Como estamos a combinar nós de baixa e alta probabilidade, reduzimos a profundidade da árvore e a sua variância.

Exercício 5

O objetivo deste exercício é repetir o anterior, recorrendo ao agrupamento de símbolos, sendo que um símbolo é na verdade uma sequência de símbolos contíguos.

Para o resolver, criámos três funções: "hist5IMG.m", "hist5SOM.m" e "hist5TXT.m", ou seja, um para cada tipo de fonte. Todos este ficheiros percorrem a fonte de modo a juntar os símbolos contíguos e devolvem uma matriz com o número de ocorrências. A função *reshape* é utilizada para colocar a matriz com apenas uma linha.

Para calcular a entropia também criamos um ficheiro diferente que é "entropia5.m", sendo que este ficheiro faz exatamente o mesmo que o ficheiro entropia que tínhamos anteriormente com a diferença de dividir por dois o tamanho da fonte, visto que juntamos os caracteres contíguos dois a dois.

Neste caso, obtivemos os resultados esperados exceto para o ficheiro "Texto.txt" que contem o mesmo erro referido no exercício 3.

Para o cálculo da entropia agrupada do ficheiro "saxriff.wav" usámos um algoritmo com a mesma lógica das outras fontes, mas que não se encontra a funcionar corretamente, por um possível erro de "alocamento de memória".

Fonte	Lena.bmp	CT1.bmp	Binaria.bmp	Saxriff.wav	Texto.txt
Entropia Agrupada	5.5965	4.4813	0.5424	ND	2.1727

Tabela 3 – Resultados da Entropia Agrupada

Exercício 6

A informação mútua é uma medida de correlação de informação que uma variável contém de outra.

Alínea a)

Nesta primeira alínea, o objetivo é calcular o vetor de calores da informação mútua em cada janela. Para isto criamos uma função ("6a.m") que tem como inputs a query, o target, o alfabeto e o step. Nesta função calculamos o número de janelas existentes, com esta informação definimos um ciclo em que calculamos o valor da informação mútua, utilizando outra função criada: "calcprobinfo.m" (o cálculo é feito através da probabilidade de cada acontecimento no respetivo conjunto e com a probabilidade conjunta dos diversos elementos.), para cada uma das janelas móveis do target. Os valores obtidos são guardados num array: "MutualInfo". Após isto tudo, a informação mútua é calculada com recurso à regra da cadeia.

Resultados:

```
Ex 6a:
  Columns 1 through 11

    2.1219    1.9219    1.6464    2.1710    1.9710    1.7710    2.0464    2.1219    2.3219    2.5219    2.2464

  Columns 12 through 22

    2.2464    2.2464    2.2464    2.4464    2.4464    2.5219    2.7219    2.5219    2.3219    2.3219    2.1219

  Columns 23 through 33

    2.3219    2.3219    2.1219    2.0464    2.0464    2.0464    2.0464    2.0464    2.3219    2.3219    2.0464

  Columns 34 through 41

    2.1219    2.1219    1.8464    1.7710    1.8464    2.0464    2.0464    2.3219
```

Alínea b)

Nesta alínea temos de determinar a variação da informação mútua entre o ficheiro "saxriff.wav" e os ficheiros "target01- repeat.wav" e "target02- repeatNoise.wav".

Em ambos os casos temos como query o "saxriff.wav".

No primeiro caso, o nosso target é "target01- repeat.wav". O gráfico mostra que o valor da informação mútua é bastante elevado nas janelas 1, 5 e 9, sendo quase nulo nas restantes, ou seja, a nossa query aparece na target 3 vezes.

O gráfico está de acordo com o esperado, visto que o target é uma repetição da nossa query.

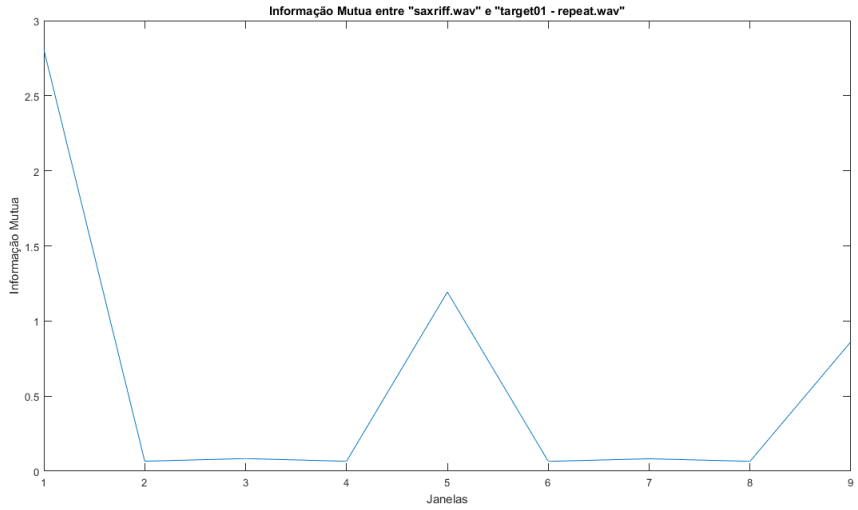

Imagem 6 – Evolução da informação mútua entre "saxriff.wav" e "target01- repeat.wav"

No segundo caso, o nosso target é "target02- repeatNoise.wav". O gráfico da informação mútua mostra valores bastantes elevados na mesmas janelas e quase nulos nas restantes. Obtemos valores mais baixos pelo ruído que está presente na primeira e última repetição.

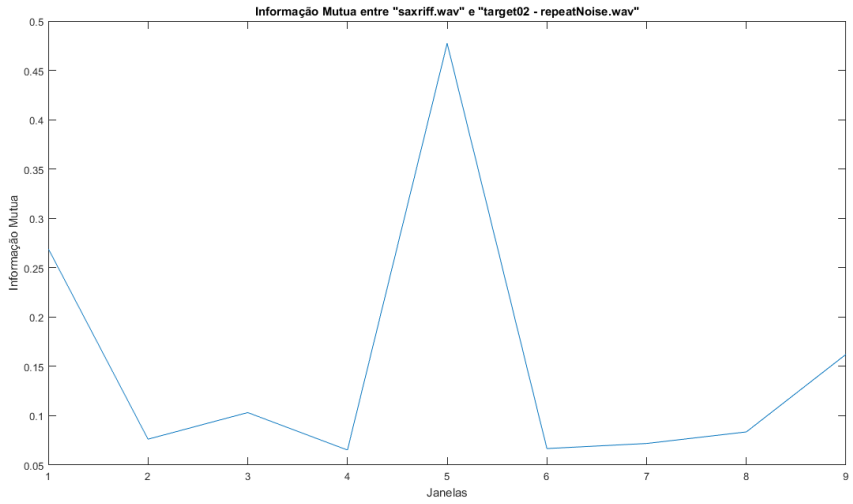

Imagem 6 – Evolução da informação mútua entre "saxriff.wav" e "target02- repeatNoise.wav

Alínea c)

Nesta alínea são nos fornecidas 7 targets diferentes. As informações mútuas são quase nulas em todos menos nos targets "Song05.wav", "Song06.wav" e "Song07.wav". No 05, temos o saxriff mas com bastante ruido, portanto a informação mútua é ligeiramente superior às anteriores. Nos ficheiros 06 e 07 temos muito mais semelhança, portanto a informação mutua é bastante maior do que todas as anteriores. Possivelmente deveria haver maior diferença entre os valores da "Song06" e "Song07" pois esta última tem menos ruído do que a anterior.

Resultados:

```
Ex 6c:
Informações Mútuas Máximas de "Song01.wav" a "Song07.wav":
    0.1417    0.1947    0.1732    0.1964    0.5369    3.5356    3.5356

Informações Mútuas Máximas ordenadas:
    3.5356    3.5356    0.5369    0.1964    0.1947    0.1732    0.1417
```