Teoria da Informação

Trabalho Prático nº 1

Entropia, Redundância e Informação Mútua

Introdução

Período de execução: 4 aulas práticas laboratoriais

Entrega:
Código completo + relatório: inforestudante

Prazo de Entrega:
21 de Outubro, sexta-feira, 23h59

Esforço extra-aulas previsto: 15h/aluno

Objectivo: Pretende-se que o aluno adquira sensibilidade para as questões fundamentais de teoria de informação, em particular informação, redundância, entropia e informação mútua.

Trabalho Prático

1. Escreva uma rotina em Matlab que dada uma fonte de informação *P* com um alfabeto $A=\{a_1,\ldots,a_n\}$ determine e visualize o **histograma de ocorrência dos seus símbolos**.

2. Escreva o código que dada uma fonte de informação *P* com um alfabeto $A=\{a_1,\ldots,a_n\}$ determine o **limite mínimo teórico para o número médio de bits por símbolo**.

3. Usando as rotinas desenvolvidas nas alíneas 1) e 2) determine a distribuição estatística e o número médio de bits por símbolo para as seguintes fontes:

- Lena.bmp
- CT1.bmp
- Binaria.bmp
- saxriff.wav
    - Nota: alfabeto de som no intervalo [-1, 1[(limite superior aberto); se o intervalo entre amostras sucessivas for 'd', o alfabeto será {-1, -1+d, -1+2d, ..., 1-d}. 'd' depende da dimensão do alfabeto (256 se 8 bits de quantização, 65536 se 16 bits, ...)
- Texto.txt (nesta fonte considere somente os símbolos regulares do alfabeto, ignorando acentos e símbolos de pontuação)

Nota: a chamada das rotinas indicadas neste ponto e nos seguintes deve ser feita num script, por exemplo, main.m

✍ Apresente os resultados.
✍ Analise e comente os resultados.
🕮 Será possível comprimir cada uma das fontes de forma não destrutiva? Se Sim, qual a compressão máxima que se consegue alcançar?Justifique.

Notas:

- A leitura de ficheiros de texto deverá ser efectuada com recurso às funções ***fopen*** e ***fscanf*** (sintaxe idêntica à da linguagem C – consultar a ajuda do Matlab em caso de dúvida)
    - Poderá também utlizar as funções ***double*** e ***num2str*** (conversão de string para número e vice-versa)

- As rotinas ***imRead*** e ***wavRead*** permitem, respectivamente, efectuar a leitura de ficheiros de imagem (bmp, gif, jpeg, etc.) e wav.
- A função ***imfinfo*** dá informação sobre uma dada imagem (e.g., dimensão, número de bits por pixel, etc.)
- As suas sintaxes são as que seguidamente se apresentam:

- ***Y = imread(filename, format)*** ou ***Y = imread(filename)***

  filename – string com o nome do ficheiro que contém a imagem a ler
  format – string com a identificação do formato do ficheiro (não especificado → inferido pela análise do ficheiro)
  Y - matriz de pixels (bidimensional para uma imagem de níveis de cinzento ou indexada e tridimensional para uma imagem RGB)

  Exemplo: A = imread('imagem.bmp')

  Para visualizar uma imagem aberta poderá usar a função ***'imshow(x)'***

  Exemplo: imshow(A)

- ***info = imfinfo(filename)***

  filename – string com o nome do ficheiro que contém a imagem cuja informação se pretende obter
  info – estrutura de dados com informação sobre a imagem

  Exemplo:

```
info = imfinfo('imagem.bmp');
disp(info.Width)
```

- ***[Y, fs, nbits] = wavread(filename)*** ou ***Y = wavread(filename)***

filename – string com o nome do ficheiro que contém o clip de áudio a ler

Y - Amostras (amplitude) do áudio; caso o áudio seja estéreo, Y conterá dois vectores de valores (amostras)

fs - Frequência de amostragem

nbits - Número de bits por amostra

Exemplo: [Y, fs, nbits] = wavread('audio.wav')

Para reproduzir um som em matlab poderá usar a função ***'wavplay(y, fs)'***

Exemplo: wavplay(Y, fs)

4. Usando as rotinas de codificação de Huffman que são fornecidas, determine o número médio de bits por símbolo para cada uma das fontes de informação usando este código.

✍ Analise e comente os resultados.
✍ Analise e comente a variância dos comprimentos dos códigos resultantes.
🕮 Será possível reduzir-se a variância? Se sim, como pode ser feito em que circunstância será útil?

Nota:
A rotina *hufflen* determina o número de bits do código Huffman necessários à codificação de um conjunto de símbolos com uma dada frequência de ocorrência. A sua sintaxe é a seguinte:

HLen= hufflen(freqOcurr)

freqOcurr = [f(1),f(2),…,f(n)] – vector com a frequência de ocorrência dos *n* símbolos em que *f*(*i*) corresponde à frequência de ocorrência do símbolo número *i* do alfabeto.

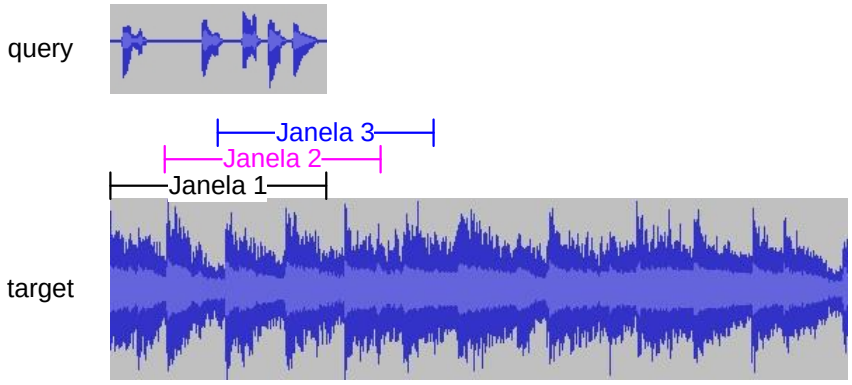
HLen = [len(1),len(2),…,len(n)] – vector com o número de bits em que *len*(*i*) representa o número de bits necessários à codificação do símbolo número i.

Exemplo:
hufflen([1,0,4,2,0,1]) => ans = [3,0,1,2,0,3]
hufflen([10,40,20,10]) => ans = [3,1,2,3]

5. Repita a alínea 3) aplicando agrupamentos de símbolos, isto é, admitindo que cada símbolo é na verdade uma sequência de dois símbolos contíguos.

✍🕮 Analise e comente os resultados.

6. Em muitas situações reais, é necessário procurar-se uma onda sonora conhecida, num sinal genérico. Nomeadamente, num contexto de comunicação em ambiente ruidoso é necessário detectar o sinal original transmitido no sinal recebido no destino, corrompido com ruído. Noutras aplicações, por exemplo em sistemas de identificação musical como o Shazam[1], é necessário procurar-se um trecho de uma música numa base de dados. Em qualquer dos casos, é possível aplicar um conjunto vasto de técnicas. Neste ponto, será calculada a **informação mútua** entre o sinal a pesquisar (*query*) e o sinal onde pesquisar (*target*), utilizando uma **janela deslizante**, de acordo com a ilustração seguinte:



Deste modo, a query será comparada com secções diferentes do target (correspondentes às janelas ilustradas). Por outras palavras, a query "deslizará" sobre o target, sendo calculada a informação mútua em cada uma das janelas. O intervalo entre janelas consecutivas é desginado por **passo**.

a) Escreva uma rotina em Matlab que, dada a query, o target, um alfabeto $A=\{a_1,\ldots,a_n\}$ e o passo, devolva o vector de valores de informação mútua em cada janela.

Simulação:
Dados:
query = [2 6 4 10 5 9 5 8 0 8]; (*vector 1x10*)
target = [6 8 9 7 2 4 9 9 4 9 1 4 8 0 1 2 2 6 3 2 0 7 4 9 5 4 8 5 2 7 8 0 7 4 8 5 7 4
3 2 2 7 3 5 2 7 4 9 9 6]; (*vector 1x50*)
alfabeto = 0 : 10;
step = 1

Resultados a obter:

[1] http://www.shazam.com; nestes sistemas, o utilizador marca o número do fornecedor de serviço no seu telemóvel, direcciona o microfone para a origem do som durante alguns segundos (de 3 a 20, dependendo do fornecedor) e aguarda uma mensagem escrita contento a identificação da música (artista, título, etc.).

infoMutua = [2.1219 1.9219 1.6464 2.1710 1.9710 1.7710 2.0464
2.1219 2.3219 2.5219 2.2464 2.2464 2.2464 2.2464 2.4464
2.4464 2.5219 2.7219 2.5219 2.3219 2.3219 2.1219 2.3219
2.3219 2.1219 2.0464 2.0464 2.0464 2.0464 2.0464 2.3219
2.3219 2.0464 2.1219 2.3219 1.8464 1.7710 2.0464 2.0464
2.0464 2.3219] (*vector 1x41*)

b) Usando o ficheiro "guitarSolo.wav" como query, determine a variação da informação mútua entre este e os ficheiros "target01 - repeat.wav" e "target02 – repeatNoise.wav". Defina um passo com valor de ¼ do comprimento do vector da query (valor arredondado).

✍ Visualize graficamente a evolução da informação mútua ao longo do tempo para cada caso.
✍ Analise e comente os resultados.

Nota: Utilize apenas o primeiro canal de cada ficheiro (primeira coluna)

c) Pretende-se agora simular um pequeno simulador de identificação de música. Usando o ficheiro "guitarSolo.wav" como query e os ficheiros Song*.wav como target:
- determine a evolução da informação mútua para cada um dos ficheiros
- calcule a informação mútua máxima em cada um deles
- e, finalmente, apresente os resultados da pesquisa, seriados por ordem decrescente de informação mútua. Defina um passo com valor de ¼ da duração da query.

✍ Apresente os resultados (informação mútua em cada caso).
✍ Analise e comente os resultados.