Teoria da Informação

Trabalho Prático nº 1

Entropia, Redundância e Informação Mútua

Introdução

Período de execução: 4 aulas práticas laboratoriais

Ritmo de execução esperado para avaliação:

- Semana 1: alíneas 1 a 3
- Semana 2: alíneas 4 e 5
- Semanas 3 e 4: alínea 6, finalização e relatório

Formato de Entrega:

Entrega final (código completo + relatório): InforEstudante

Prazo de Entrega:

9 de Novembro, 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 função 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 funções desenvolvidas nas alíneas 1) e 2), determine a distribuição estatística (histograma) e o limite mínimo para o número médio de bits por símbolo das seguintes fontes:

- lena.bmp
- homer.bmp
- binaria.bmp
- saxriff.wav
- texto.txt (nesta fonte considere somente os símbolos regulares do alfabeto, ignorando símbolos da fonte que contenham acentos ou sinais de pontuação)

✍ 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 ***open*** e ***read*** (consultar a ajuda do Python em caso de dúvida)
    - Poderá também utilizar a função ***numpy.asarray*** (conversão de *list* para array do numpy)

4. Usando as funções 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:
O método *get_code_len* da classe *HuffmanCodec* do módulo *huffmancodec* 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:

codec = HuffmanCodec.from_data(P)
symbols, lenghts = codec.get_code_len()

P – fonte de informação, e.g., matriz representativa de uma imagem.

symbols – array com os símbolos presentes em P. Nota: poderá ser um sub-conjunto do alfabeto completo da fonte.

lengths – array com o comprimento respectivo de cada um dos símbolos de P.

Exemplo: ver função main do ficheiro huffmancodec.py

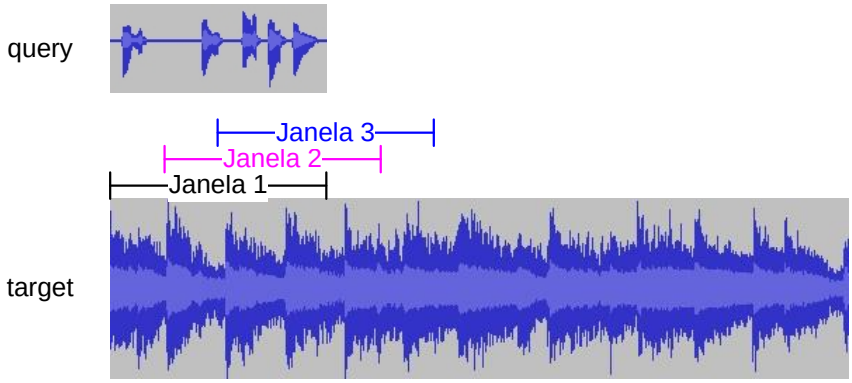
5. Repita a alínea 3) aplicando agrupamentos de símbolos, isto é, admitindo que cada símbolo é na verdade uma sequência de dois símbolos contíguos.

✍🕮 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

[1] http://www.shazam.com; nestes sistemas, o utilizador direcciona o microfone do telemóvel para a origem do som durante alguns segundos e a aplicação identifica a música correspondente (artista, título, etc.).

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 função 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]; (*array 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]; (*array 1x50*)
alfabeto = 0, 1, 2, …, 10;
passo = 1

Resultados a obter:
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] (*array 1x41*)

b) Usando o ficheiro "saxriff.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 target.
✍ Analise e comente os resultados.

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

c) Pretende-se agora criar um pequeno simulador de identificação de música. Usando o ficheiro "saxriff.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.