Project 2

Information Theory 2022

Cada aluno deverá escrever (individualmente) um módulo Python 3 `ldpc.py`, que permite a codificação e decodificação de mensagens usando códigos LDPC. O módulo deve implementar três funções: `generate_code(K, eps)`, `encode(K, P, w)` e `decode(K, P, y)`, com a seguinte especificação :

- A função `generate_code(K, eps)` aceita um inteiro não negativo K, o comprimento da mensagem e um parâmetro float `eps`, e deve gerar (usando aleatoriedade) o código LDPC que foi ensinado na aula. O código é representado por uma lista `P` de tamanho aproximado $\approx \frac{\varepsilon K}{1-\varepsilon} = K(\varepsilon + \varepsilon^2 + \varepsilon^3 + \dots)$, onde cada elemento da lista `P` é uma lista de números entre 0 e $K-1$. A lista `P[j]` contém os índices dos bits cuja paridade deve ser o bit número $K+j$ da mensagem codificada; os primeiros K bits da mensagem codificada são obviamente a própria mensagem.

  O número de paridades *é permitido* ser maior que $\frac{\varepsilon K}{1-\varepsilon}$. O seu objetivo é escolher os bits de paridade para que esse número seja o mais próximo possível de $\frac{\varepsilon K}{1-\varepsilon}$, pelo menos à medida que K cresce.

- A função `encode(K,P,w)` aceita um número K, um código LDPC P e uma mensagem w, que para os propósitos desta atribuição é representada como uma string de caracteres `0` e `1`.

  O output deve ser outra string de caracteres `0` e `1`, de tamanho $K + |P|$, consistindo de w concatenado com bits de paridade $|P|$, correspondendo à codificação de mensagem w pelo código P.

- A função `decode(K, P, y)` aceita K e P como antes, junto com uma string com os caracteres `0`, `1`, e `?`.

  A função `decode(K, P, y)` deve retornar uma string de caracteres `0` e `1`, de tamanho K, que é a melhor estimativa do algoritmo para qual mensagem w foi codificado por P e enviado pelo canal. Se a decodificação falhou, a função deve retornar `None`. É recomendável que você use o decodificador simples ensinado em sala de aula, mas você pode usar qualquer outro método à sua preferência (o método ideal resolve o sistema implícito de equações lineares sobre $\mathbb{F}_2$, mas isso é muito caro em tempo).

Note o seguinte:

- Para gerar subconjuntos aleatórios de um determinado tamanho, você pode usar a função `sample` do módulo Python 3 `random`. Por exemplo. `sample(range(10), 3)` escolhe 3 números aleatórios únicos entre 0 e 9.

- Para testar seu código, o aluno também deve implementar uma função, `transmit(x, eps)`, que retorna uma string igual a x mas onde cada caractere é substituído por `?` por probabilidade `eps`. Isso pode ser feito usando a função `random` do módulo `random`, que retorna um número entre `0.0` e `1.0`, e comparando esse número com `eps`.

  Deve então acontecer que

  ```
  decode(K,P,transmit(encode(K,P,w),eps)) == w
  ```

  com probabilidade tendendo a 1 conforme K cresce. Se isso ocorrer (durante a avaliação, utilizarei as minhas próprias implementações de `encode` e `transmit`), e o comprimento $|P|$ estiver dentro de um fator de 10 de $\frac{\varepsilon K}{1-\varepsilon}$, então o aluno receberá 3 pontos. Se estiver dentro de um fator de 3, o aluno receberá 4 pontos. Se estiver dentro de um fator de 1.5, o aluno receberá 4 pontos mais meio ponto de bônus. Se o número de paridades $|P|$ se aproximar do ideal $\frac{\varepsilon K}{1-\varepsilon}$ à medida que K cresce, o aluno receberá 4 pontos mais um ponto de bônus. Se o aluno incluir uma prova de que esse é sempre o caso, ele receberá 4 pontos mais 4 pontos de bônus.