TP555 - AI/ML
Lista de Exercícios #0

Instruções

- **Siga atentamente todas as instruções.**
- Utilizaremos notebooks Jupyter para resolver os exercícios desta e das demais listas. Os notebooks podem ser executados localmente, via binder, ou na nuvem, via google colab.
- Siga as instruções do link a seguir caso você queira instalar o binder em seu computador pessoal, certifique-se de instalar a versão 3.x do Python e não a versão 2.x pois esta foi descontinuada:

  https://docs.anaconda.com/anaconda/install/
- Você pode instalar o Jupyter e os softwares relacionados seguindo os passos do tutorial abaixo:

  https://drive.google.com/file/d/1AORykPraAlDC_wTOgjM_aAzVJ11IF29P/view
- Para executar os notebooks na nuvem, via Colab, acesse:

  https://colab.research.google.com/
- Os 2 links seguintes contém tutoriais sobre os notebooks Jupyter:
  - https://jupyter-notebook.readthedocs.io/en/stable/notebook.html
  - https://jupyter-notebook.readthedocs.io/en/stable/ui_components.html
- **Crie um notebook Jupyter para cada exercício das listas**
- Para armazenar as listas, provas e trabalho final, você precisá criar um repositório no GitHub.
- Veja as várias referências adicionadas ao final deste documento para entender os conceitos básicos de Git, GitHub e Python.
- Seu repositório **DEVE** ser **privado** sendo o professor a única outra pessoa a ter acesso a ele. Leia as instruções no exercício 1 abaixo para entender como fazer isso. **OBS.**: meu usuário no github é **luiz10ml**, por favor, **não compartilhe** o acesso ao seu repositório com o meu email do Inatel.
- A biblioteca Numpy será de extrema importância para nosso curso. Você pode aprender como utilizá-la através do vídeo citado na referência [8].
- Para os exercícios envolvendo modulação e demodulação, 7, 8 e 9, por favor, veja os links presentes na referência [9].

Exercícios

1. Crie um repositório **privado** no github com seu nome seguido de seu número de matrícula mais a palavra tp555, exemplo: luiz-20-tp555. Em seguida, me dê acesso ao seu repositório **privado.** **OBS.**: meu usuário no github é luiz10ml, por favor, não compartilhem o acesso ao repositório com o meu email do Inatel. Este repositório servirá para que você versione seus exercícios/trabalhos e os entregue para serem avaliados. Dentro do repositório, crie uma pasta com o nome **lista0** e dentro desta pasta salve os notebooks com os códigos dos exercícios. Faça o mesmo para todas as outras listas de exercícios. Ao finalizar a lista, me envie o link para o seu repositório. (**Dica:** Não crie um repositório para cada lista que você for resolver durante o curso, você deverá ter apenas um repositório, mas haverão várias pastas, nomeadas com os números das listas, fazendo parte do seu repositório. Veja as referências [1],[2],[3] e [4] para aprender ou relembrar como usar o Git e GitHub.) (**Dica:** Para tornar seu repositório privado e me dar acesso à ele, veja o documento TP555_Private_Repository.pdf que também está disponível no MS Teams.)

2. Execute cada um dos exemplos do documento: Exemplos de códigos em Python.pdf. Para isso, crie um notebook Jupyter diferente para cada um deles. Nomeie cada um dos notebooks Jupyter com o nome que consta no documento. Caso você tenha instalado o Jupyter/Binder localmente, no Windows, digite Jupyter na barra de buscas e selecione `Jupyter Notebook` para iniciar a aplicação. No Linux, abra um terminal e digite jupyter notebook. Outra forma de encontrar o aplicativo no Windows é ir até o menu Iniciar, encontrar o Anaconda e escolher a opção Jupyter Notebook. Caso você queira resolver o exercício através das aplicações web, acesse os links dos projetos que se encontram nos slides de introdução.

3. Neste exercício você irá plotar um gráfico 2D. Este tipo de gráfico é comumente utilizado para se analisar os dados de entrada e saída de um modelo de aprendizado de máquina. Crie um vetor coluna, $\mathbf{y}$, com $M = 1000$ elementos, onde y é dado pela seguinte equação

$$y = 1,2 + 2,3x + 10w$$

onde x é um vetor coluna com M elementos retirados de uma distribuição aleatória uniforme com valores no intervalo em $[0, 1)$ e w é um vetor coluna com M elementos retirados de uma distribuição aleatória Gaussiana normal, i.e., com média 0 e variância unitária. Plote um gráfico com os vetores $\mathbf{x}$ e $\mathbf{y}$ sendo os eixos x e y, respectivamente. Cada par de valores (x,y) deve ser mostrado no gráfico como sendo um ponto.

Dicas:

- Use o módulo random da biblioteca numpy para gerar números aleatórios. https://numpy.org/doc/stable/
- Use a biblioteca matplotlib para plotar gráficos. https://matplotlib.org/stable/gallery/lines_bars_and_markers/simple_plot.html

4. Neste exercício você vai plotar o histograma de um vetor criado através da soma de variáveis aleatórias. Histogramas são utilizados para se verificar a distribuição de um determinado conjunto de dados. Crie um vetor coluna x com $M = 10000$ amostras retiradas de uma distribuição aleatória uniforme. Em seguida, crie outro vetor coluna y, também com $M = 10000$ amostras retiradas de uma distribuição aleatória uniforme. Na sequência, obtenha o vetor z, que é definido pela seguinte equação

$$z = x + y$$

Plote o histograma normalizado de z

Dicas:

- Use o método hist da biblioteca matplotlib.
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html
https://matplotlib.org/stable/gallery/statistics/histogram_normalization.html

5. Neste exercício você irá plotar um gráfico 3D. Este tipo de gráfico pode ser utilizado para visualizar superfícies de erro, as quais são comumente encontradas em problemas de otimização. Crie 2 vetores, $\mathbf{x}_1$ e $\mathbf{x}_2$, respectivamente, com valores uniformemente espaçados entre -10 e 11 com passos de 0,25 unidades. Em seguida crie o vetor y, o qual é definido pela seguinte equação

$$y = x_1^2 + x_2^2$$

Plote um gráfico 3D com x_1, x_2 e y sendo plotados nos eixos x, y e z, respectivamente.

Dicas:

- Use o método arange da biblioteca numpy para gerar valores uniformemente espaçados com passos predefinidos..
https://docs.scipy.org/doc/numpy/reference/generated/numpy.arange.html
- Estude o seguinte exemplo para entender como plotar gráficos 3D
https://matplotlib.org/stable/gallery/mplot3d/surface3d.html

6. Usando a função numpy.array crie 2 arrays 1D (uma dimensão) com os seguintes valores 0,1,0,1 e 0,0,1,1, respectivamente. Em seguida, concatene com a função numpy.c_ esses dois vetores em uma matriz 2D com dimensão 4×2. Imprima a dimensão dessa matriz através do atributo shape da matriz criada. Em seguida, imprima o conteúdo da matriz resultante da concatenação.

Dicas:

- A documentação da função numpy.array pode ser acessada através:
https://numpy.org/doc/stable/reference/generated/numpy.array.html

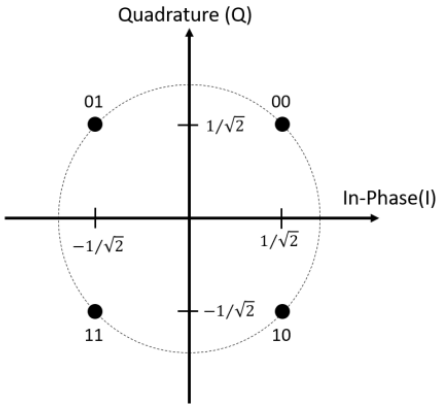
- A documentação da função numpy.c_ pode ser acessada através: https://numpy.org/doc/stable/reference/generated/numpy.c_.html

7. Implemente uma **função** chamada **modulator** que receba como parâmetro de entrada um valor de 0 a 3 e retorne um dos seguintes números complexos: $[1/\sqrt{2} + j1/\sqrt{2},\ -1/\sqrt{2} + j1/\sqrt{2},\ 1/\sqrt{2} - j1/\sqrt{2},\ -1/\sqrt{2} - j1/\sqrt{2}]$. Em seguida, de posse da função já implementada, crie uma array com $N = 1000$ valores aleatórios, variando entre 0 e 3, com a função **numpy.random.randint**, passe esse vetor para a **função modulator** e armazena a saída da função em um vetor symbols. Plote a constelação resultante da modulação, ou seja, utilize o vetor symbols.

   Dica:

   - A documentação da função numpy.random.randint pode ser encontrada em: https://numpy.org/doc/stable/reference/random/generated/numpy.random.randint.html

8. Implemente uma função, chamada **demodulator**, que receba um vetor com números complexos da forma mostrada na tabela abaixo na coluna **Símbolo** e retorne um vetor com os valores decimais de cada um dos pares de bits. Em seguida, de posse da função implementada, use a saída gerada pela função **modulator** do exercício anterior como entrada para a função **demodulator** e compare com os valores aleatórios variando de 0 a 3 que você usou como entrada para a função modulator. Observe que não pode haver nenhum erro, ou seja, como não há ruído adicionado ao sinal modulado, seu demodulador irá recuperar todos os bits transmitidos perfeitamente



bits	decimal	Símbolo (I + jQ)
00	0	$\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$
01	1	$-\frac{1}{\sqrt{2}} + j\frac{1}{\sqrt{2}}$
10	2	$\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$
11	3	$-\frac{1}{\sqrt{2}} - j\frac{1}{\sqrt{2}}$

9. Agora, com as duas funções implementadas nos 2 exercícios anteriores, faça o seguinte:

   (a) Crie uma array com $N = 1000000$ valores aleatórios, variando entre 0 e 3, com a função **numpy.random.randint**, passe esse vetor para a função modulator e armazena a saída da função em um vetor symbols.

(b) Adicione ruído gaussiano branco ao vetor de saída da função modulator. Varie a relação energia de símbolo (Es) por densidade espectral do ruído (N0) de -2 a 20 dB em passos de 2 dB.

(c) Usando a função **demodulator**, calcule o erro de símbolo simulado para cada valor de Es/N0.

(d) Em seguida, plote um gráfico comparando o taxa de erro de símbolo (SER) simulado com a taxa de erro de símbolo teórica, a qual é dada por

$$\text{SER} = \text{erfc}\left(\sqrt{\frac{Es}{2N_0}}\right) - \frac{1}{4}\text{erfc}\left(\sqrt{\frac{Es}{2N_0}}\right)^2$$

Dica:

- As duas curvas devem coincidir quase que perfeitamente

Referências

[1] An Intro to Git and GitHub for Beginners (Tutorial):
https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners

[2] GitHub: Hello World:
https://guides.github.com/activities/hello-world/

[3] Como usar Git e Github na prática: Guia para iniciantes:
https://www.youtube.com/watch?v=2alg7MQ6_sI

[4] APRENDA GIT e GITHUB DO ZERO - guia completo:
https://www.youtube.com/watch?v=pyM5QLS2h6M

[5] Python.org, "BeginnersGuide":
https://wiki.python.org/moin/BeginnersGuide/Programmers

[6] Mark Pilgrim, "Dive into Python":
https://diveintopython3.problemsolving.io/

[7] Nerd Paradise, "4 Minute Python Crash Course":
https://nerdparadise.com/programming/python4minutes/

[8] Didática Tech, 'Aprenda como usar o Numpy (Python para machine learning - Aula 10)':
https://www.youtube.com/watch?v=CC4aco6zWic

[9] Alguns exemplos que podem ajudar com os exercícios envolvendo modulação/demodulação:
https://scipy.github.io/old-wiki/pages/Cookbook/CommTheory.html
https://inst.eecs.berkeley.edu/~ee123/sp15/lab/lab6/Pre-Lab6-Intro-to-Digital-Communications.html

http://pysdr.org/content/digital_modulation.html
http://www.raymaps.com/index.php/bpsk-bit-error-rate-calculation-using-python