TP555 - AI/ML

Lista de Exercícios #0

Instruções

- Utilize o Jupyter para resolver os exercícios desta lista. Siga as instruções do link a seguir caso você queira instalá-lo em seu computador pessoal: https://docs.anaconda.com/anaconda/install/
- Crie um notebook do Jupyter para cada exercício da lista. Faça isto para todas as outras listas que serão disponibilizadas.
- Por favor, me enviem um email avisando que você terminou cada uma das listas que irei disponibilizar ao longo do curso. Eu mantenho uma planilha com as listas finalizadas de todos os alunos.
- Vejam as várias referências adicionadas no final deste documento para vocês entenderem conceitos básicos de Git, GitHub e Python.
- Seu repositório DEVE ser **privado** sendo eu a única outra pessoa a ter acesso. Leia as instruções no exercício abaixo para ver como fazer isso.
- A biblioteca Numpy será de extrema importância para nosso curso. Você pode aprender como utilizá-la através do vídeo citado na referência [8].

Exercícios

1. Crie um repositório ***privado*** no github com seu nome seguido de seu número de matrícula mais a palavra tp555. Em seguida me dê acesso ao seu repositório ***privado***. Exemplo: felipe-131-tp555. 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 sendo 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 está disponível no site.)
2. Execute cada um dos exemplos dos slides. Crie um notebook Jupyter para cada um deles. No Windows, digite Jupyter na barra de buscas e selecione Jupyter Notebook. No Linux, abra um terminal e digite `jupyter notebook`. Já no Windows, vá até o menu Iniciar, encontre o Anaconda e escolha a opção Jupyter Notebook.
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, y, com M = 1000 elementos, onde y é dado pela seguinte equação

$$y = 1.2 + 2.3*x + 10*w,$$

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.

a. Plote um gráfico com os vetores x e 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://docs.scipy.org/doc/numpy-1.14.0/reference/routines.random.html
- Use a biblioteca matplotlib para plotar gráficos https://matplotlib.org/3.1.3/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

$$z = x + y.$$

a. Plote o histograma normalizado de z.

Dica:

- Use o método hist da biblioteca matplotlib https://matplotlib.org/3.1.3/api/_as_gen/matplotlib.pyplot.hist.html https://matplotlib.org/3.1.3/gallery/statistics/histogram_features.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, x1 e x2, 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

$$y = x1^2 + x2^2.$$

a. Plote um gráfico 3D com x1, x2 e y sendo plotados nos eixos x, y e z, respectivamente.

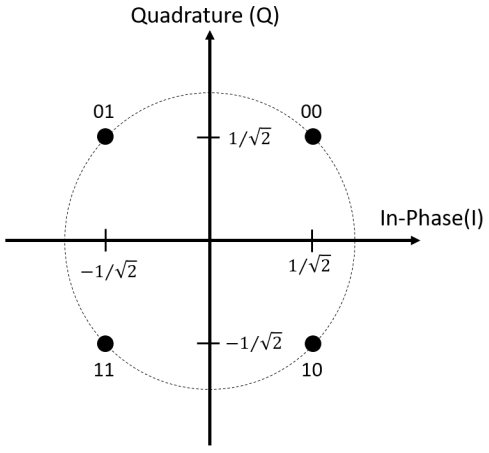
Dica:

- 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/3.1.3/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 matrix 2D com dimensão 4x2. Imprima a

dimensão dessa matriz através do ***atributo shape*** da matriz criada. Em seguida, imprima o conteúdo da matriz resultante da concatenação.
(**Dica**: A documentação da função numpy.array pode ser acessada através: https://numpy.org/doc/1.18/reference/generated/numpy.array.html)
(**Dica**: A documentação da função numpy.c_ pode ser acessada através: https://numpy.org/doc/1.18/reference/generated/numpy.c_.html?highlight=numpy%20c_#numpy.c_)

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://docs.scipy.org/doc/numpy-1.15.1/reference/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

$$\text{SER} = erfc\left(\sqrt{\frac{Es}{2N0}}\right) - \frac{1}{4}erfc\left(\sqrt{\frac{Es}{2N0}}\right)^2$$

(**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] 'Guia Completo do Iniciante: Git e GitHub', https://www.youtube.com/watch?v=UbJLOn1PAKw
[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&list=PLyqOvdQmGdTR46HUxDA6Ymv4DGsIjvTQ-&index=10