Observatório do Valongo / UFRJ
Métodos Computacionais da Astronomia 2020/1-PLE

Professor: Pedro da Silveira Ferreira

Estudo dirigido / Lista 2 parte 2 (5 pontos)

Entrega: 25/09/2020

---

1. [Numpy – Índices e estruturas] Resolva os itens **b)** até **g)** utilizando as propriedades da indexação dos arrays. **a)** Leia o arquivo array.dat usando o comando np.loadtxt() e armazene o mesmo como "array_original" **b)** Defina um novo array como sendo igual a 3ª coluna do array_original, e um outro array como sendo a 5ª linha do array_original. **c)** Defina um novo array com com as colunas pares do array_ogirinal **d)** Defina um novo array com as linhas impares do array_original **e)** Defina um novo array com os números múltiplos de 5 do array_original. **f)** Defina um array com apenas os números positivos do array_original. **g)** Defina um novo array como sendo a transposta do array_original **h)** Defina um novo array como sendo um array achatado (apenas uma linha) do array_original usando reshape, ravel ou flatten (olhe na documentação do numpy imbutida nos links anteriores) **i)** Utilizando reshape sobre o array do item anterior, gere um array final com 12 linhas e 2 colunas. (1 ponto)

2. [Numpy – Funções] Resolva os itens a seguir utilizando as funções do numpy **a)** Calcule a média, desvio padrão, quadrado, raiz quadrada, seno, soma (dos itens de cada array) e soma acumulada (dos itens de cada array) dos arrays gerados na questão 1. **b)** usando dois arrays aleatórios criados com np.random.random(size=50), faça as operações de soma, divisão, multiplicação e produto escalar entre os arrays. (0.5 ponto)

3. [For com list comprehension] Resolva utilizando as funções do numpy e/ou math. Gere uma lista de valores da função seno variando de -2π até 2π em passos de $\pi/100$ (use a função np.arange para definir o range do for e o pi do np.pi ou math.pi) utilizando list comprehension. (0.5 ponto)

4. [Funções + Pandas + Loops aninhados + Numpy] – **Medindo a constante de Hubble**
a) Utilizando o pandas, leia a tabela diagrama_de_Hubble.csv, que contém as velocidades de recessão e distâncias de diversas galáxias **b)** leia a mesma tabela com o numpy usando a função np.genfromtxt com o delimitador correto (geralmente ";","," "\t" ou '\s+' em arquivos csv, veja a documentação para mais detalhes), perceba que para tabelas simples como essa o numpy já basta. **c)** Faça um plot dos pontos da tabela utilizando o matplotlib. **d)** Utilizando o método dos mínimos quadrados* (MMQ) obtenha a constante de Hubble de expansão do universo ajustando uma reta aos pontos (veja a referência do método em referência). Utilize MMQ via força bruta (testando diversos valores) com um loop for executado em um intervalo razoável (após inspeção do plot do item anterior) para os valores de a e b da reta, esse loop deve ser feito de forma que primeiro ocorra em passos mais largos (primeira iteração), encontre 3 pontos do intervalo com o menor erro e novamente subdivida esse intervalo em passos menores (próxima iteração), isso deve ocorrer o número de vezes o suficiente para que o a e b variem menos que 0.2% entre uma iteração e outra (múltiplas iterações). Crie a função que aplica o mmq a ser utilizada nos loops. **e)** Faça também o MMQ na forma analítica (utilizando as fórmulas analíticas fornecidas nos comentários, as linhas sobre os símbolos representam a média daquele valor), utilize as funções praticadas no exercício 2. No caso da linha reta o MMQ possui uma solução analítica exata, porém para casos mais complexos não, é necessário um código que utilize múltiplas iterações como feito no item d). Você encontrou um valor próximo ao valor encontrado na literatura para a constante de Hubble? (Referência para os valores) (1.5 pontos)

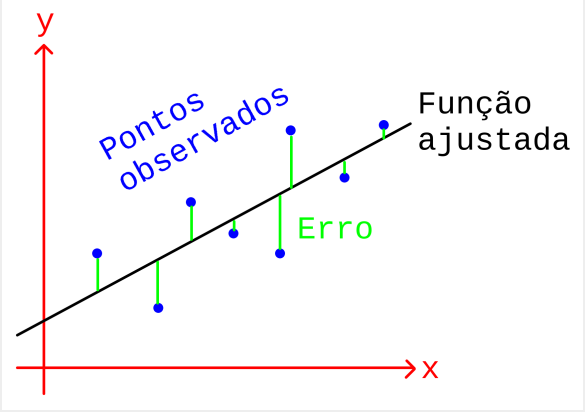
5. [Argparse + Astropy] **– Posição de astros em tempo real e mudança de coordenadas**
a) Utilizando o argparse e o astropy, faça um código em python que peça as coordenadas do observador (latitude, longitude e altitude) e o nome do objeto e retorne (print) as coordenadas do objeto no tempo atual (é livre a escolha da biblioteca para obter o tempo), em coordenas altazimutais e equatoriais. Veja este tutorial e este tutorial para auxiliar com o astropy. **b)(extra)** Utilize o comando watch no terminal para que as coordenadas sejam atualizadas de 2 em 2 segundos, experimente com as coordenadas da lua. (1.25+0.25 pontos)

***Comentários:**

O MMQ é baseado na minimização dos quadrados dos erro (diferença entre os pontos observados e a reta). Na figura ao lado os erros são representados em verde. A equação abaixo é a forma "força bruta" do MMQ para a reta (que será a função ajustada).

$$f(x) = \sum_{pontos\ obs} \left(y^{observador} - y^{ajustado} \right)^2$$

$$f(x) = \sum_{pontos\ obs} \left(y^{observador} - a\,x - b \right)^2$$



As equações abaixo representam a forma analítica, exata, para o caso da reta:

$$a = \bar{y} - b\bar{x}$$

$$b = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Esse é o método mais simples de ajuste de uma reta.