25 Aula 25: 07/NOV/2019

25.1 Aula passada

- paradoxo do aniversário
- colecionador de figurinhas

25.2 Hoje

Mais modelos computacionais e simulação em pesquisa científica:

- área círculo/pi
- máquina de votação

Todos os problema tem um esqueleto idêntico e isso talvez valha a pena ser obervado.

Todos os diretórios têm um arquivo `main.py` que o cliente da classe principal:

- `Pi` (`pi.py`) para área do círculo
- `Votacao` (`estimativa.py`) para máquina de votação

Todos os diretórios tem um arquivo `config.py` com constantes para os programas. As constantes poderiam ser parâmetros.

Método científico

Está presente em tudo que faremos.

- **observar** algum aspecto da natureza
- **hipotetizar** um modelo consistente com as observações
- **prever** eventos usando as hipóteses
- **verificar** as predições fazendo mais observações
- **validar** repetindo até que hipoteses e obeservações estajam de acordo

25.3 Área círculo

Suponha que não sabemos a fórmula da área do círculo de raio `r`. Utilizaremso o método de Monte Carlo para determinar área do círculo. Isso serve para validar este arcabouço.

```
from config import *
import random

class Pi:
    #----------------------------------------
    def __init__(self, n, t = T, r = 1):
        self.n = n    # número de pontos
        self.t = t    # número de testes
        self.raio = r # raio do círculo
        area_semicirculo = 0
        for i in range(t):
            area_semicirculo += self.experimento()
        self.p = 4*area_semicirculo/t

    #----------------------------------------
    def mean(self):
        return self.p

    #----------------------------------------
    def experimento(self):
        ''' Sorteia um ponto dentro do círculo
        '''
        n = self.n
        r = self.raio
        if n == 0: return 0
        cont = 0
        for i in range(n):
            x = random.random()*r
            y = random.random()*r
            if x*x + y*y < r*r:
               cont += 1
        return r*r*(cont/n) # probabilidade de cair no semicírculo.
```

Experimentos

Esperimentos sugerem que a medida que o raio dobra a área do círculo é multiplicada por 4.

raio	área experimental	comando
1	3.141718750000016	python main.py 100 1
2	12.575898437500069	python main.py 100 2
4	50.2770312500003	python main.py 100 4
8	200.752500000001	python main.py 100 8
16	804.2175000000044	python main.py 100 16

Método científico

Hipótese. A área do círculo tem a forma $\texttt{a}\texttt{r}^{\texttt{b}}$ para constantes `a` e `b`.

Fazendo o gráfico log área por log `r` (bilog) se obtemos uma reta temos que o coeficiente angular é o valor de `b`, que no caso é 2.

Tem o valor de $\texttt{b} = 2$ podemos determinar o valor da constante `a` usando a hipótese, o valor de `b` e os resultados experimentais.

$$a = \text{área}/r^2 = 804.2175000000044/16^2 = 3.141474609375017$$

Logo, aproximando o valor de `a` obtemos uma aproximação de π.

25.4 Máquinas de votação

Máquina de votação. Suponha que em uma população de 100 milhões de eleitores, 51% votam para a(o) candidata(o) A e 49% votam para a(o) candidata(o) B. Entretanto, a máquina de votação é propensa a cometer erros e 5% das vezes ela contabiliza um voto de maneira errada.

Supondo que os erros ocorrem uniformemente ao acaso, 5% é uma taxa de erro suficiente para invalidar a eleição? Qual a taxa de erro que pode ser tolerada? Bem, podemos olhar esse problema da perspectiva estatística. Há certamente outras perpectivas relevantes.

Modelagem

Suponha que `no_votos` é o número total de votos, `por_A` é a porcentagem do eleitos que votaram no candidadto `A` (valor entre 0 e 100). Portanto a porcentagem do eleitores de `B` é `por_B = 100 - por_A`. Suponha ainda que `por_erro` é a porcentagem de votos errados.

No arquivo `config.py` temos as definicões

```
NO_VOTOS  = 10**8 # 100 milhões
POR_A     = 51    # porcentagem de eleitores de A
POR_B     = 49    # porcentagem de eleitores de B
POR_ERRO  = 5     # porcentagem de erro
T         = 100   # número de trials
```

Poderíamos calcular o número esperado `n_erros` de votos errados e sortear esses `n_erros` votos. Se o voto pertence ao candidato `A` o voto vai para o candidado `B` e se o voto pertence ao candidato `B` ele vai para o candidado `A`. Alternativamente, poderíamos, para cada voto, realizar um sorteio para cada voto a fim de decidir se ele iria para o outra(o) candidata(o).

Para selecionar o voto a ser alterado basta selecionarmos um inteiro `voto` aleatoriamente em 'range(no_votos). `Após selecionarmos devemos fazer`no_votos -= 1‘.

Se `votos_A` é o número de votos no candito `A`, então consideramo que o `voto` sorteado é do candidato `A` se `voto in range(votos_A)` e do candidato `B` em caso contrário.

Cliente

Cliente da classe `Votacao` que está logo a seguir.

```
def main():
    no_votos  = int(...)
    por_A     = float(...)
    por_erro = float(...)
    votacao = Votacao(no_votos, por_A, por_erro, t)
    print("prob mudar resultado = %g"votacao.mean()
```

Classe `Votacao`

```
class Votacao:
    def __init__(self, no_votos, por_A, por_erro, t):
        '''(Votacao, int, float, float, int) ->

        Recebe

             - no_votos: número de votos
             - por_A: porcentagem de votos para o candidato A
             - por_erro: porcentagem de votos errados
             - t: número de experimentos
        determina a probabilidade do resultado da eleição ser alterado
        '''
        self.no_votos = no_votos
        self.por_A = por_A
        self.por_erro = por_erro

        # realize os t experimentos
        alterou = 0
        for i in range(t):
           alterou += experimento(p_erro)
        self.p = t/alterou

    #-----------------------------------------------------------
    def mean(self):
        return self.p
```

```
#-------------------------------------------------------------
def experimento(self):
    '''(Votacao) -> bool

    Recebe a porcentagem de votos que são contados de forma errada
    e retorna True se o resultado da eleição foi alterado e False
    em caso contrário.

    Supõe que a porcentagem POR_A de eleitores de A é
    maior que porcentagem POR_B de eleitores de B.
    '''
    no_votos = self.no_votos
    por_erro = self.por_erro
    por_A    = self.por_A

    # calcule número de votos em A e B
    votos_A  = (no_votos * por_A) // 100
    votos_B  =  no_votos - votos_A
    p_erro = por_erro/100

    vA = 0
    for i in range(votos_A):
        vA += random.random() > p_erro

    for i in range(votos_B):
        vA += random.random() < p_erro

    vB = no_votos - vA
    return  vA <= vB
```

Outro versão

```
    #-------------------------------------------------------------
    def experimento(self):
        '''(int) -> bool

        Recebe a porcentagem de votos que são contados de forma errada
        e retorna True se o resultado da eleição foi alterado e False
        em caso contrário.

        Supõe que a porcentagem PORCEN_A de eleitores de A é
        maior que porcentagem PORCEN_B de eleitores de B.
        '''
        no_votos   = self.no_votos

        # calcule número de votos errados
        n_erros = (no_votos * self.por_erro) // 100

        # calcule número de votos em A e em B
        votos_A  = (no_votos * self.por_A) // 100

        # print(votos, n_erros, limiar)
        # número líquido de votos de B para A
        muda = 0

        # supor que os votos de A estão em range(votos_A)
        # supor que os votos de B estão em range(voto_A, no_votos)
        for i in range(n_erros):
            # escolhe um voto para alterarmos o resultado
            voto = random.randrange(no_votos)
            if voto < votos_A:
                # transfere um voto de A para B
                muda -= 1
                # diminui os votos de A que podem ser transferidos
                votos_A -= 1
            else:
                # transfere um voto de B para A
                muda += 1
            no_votos -= 1

        votos_A = (self.no_votos * por_A) // 100
        votos_B =  votos - votos_A

        return  votos_A + muda <= votos_B - muda
```

25.5 Apêndice

Lei dos grandes números

A **lei dos grandes números** diz que a média aritmética dos resultados da realização da mesma experiência repetidas vezes tende a se aproximar do valor esperado à medida que mais tentativas se sucederem. Em outras palavras, quanto mais tentativas são realizadas, mais a probabilidade da média aritmética dos resultados observados irá se aproximar da probabilidade real.

Método de Monte Carlo

O método de Monte Carlo é uma classe de algoritmos computacionais que se baseiam em repetições amostragens aleatórias para se obter um resultado numérido. A ideia central é usar aleatoriedade para resolver problemas que podem ser determinísticos em princípio. O método é frequentemente utilizado em física e matemática quando se é difícil ou impossível de se utilizar outros métodos. O método de Monte Carlo é usado principalmente em três classes de problemas: otimização, integração numérica e geração de sorteios de uma dada distribuição de probabilidades.