DRUNKARD'S WALK

CAMINHADA DO BÊBADO

INTEGRANTES

ISAQUE ARAÚJO NOGUEIRA;

JHONATAN DE SOUSA CARVALHO;

JONNAS CHRISTIAN SOUSA DE PAIVA;

PAULO VICTOR BRAGA CASTRO.

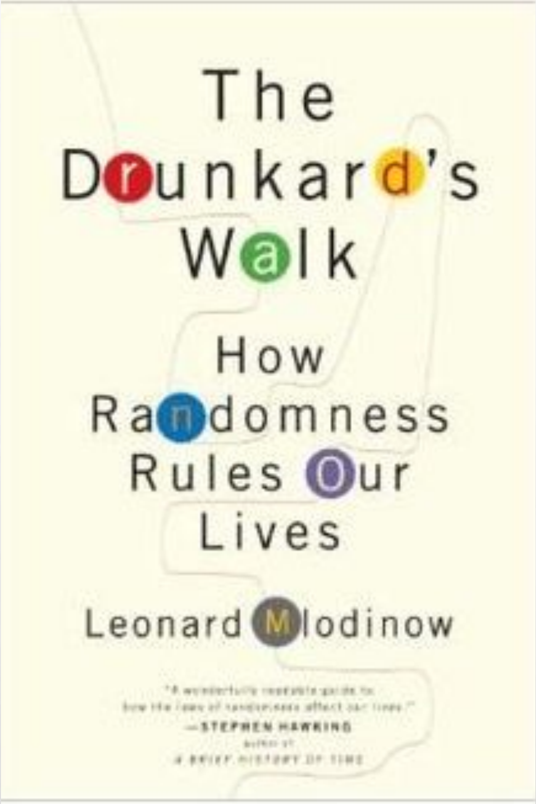
CONTEÚDO PROGRAMÁTICO DA AULA

1. CURIOSIDADES;
2. INTRODUÇÃO;
3. FUNDAMENTAÇÃO TEÓRICA;
4. METODOLOGIA E RESULTADOS;
5. SIMULAÇÃO.

1. CURIOSIDADES

1.1 O QUE É DRUNKARD'S WALK?

- **LEONARD MLODINOW;**
- **ALEATORIEDADE EM NOSSAS VIDAS.**



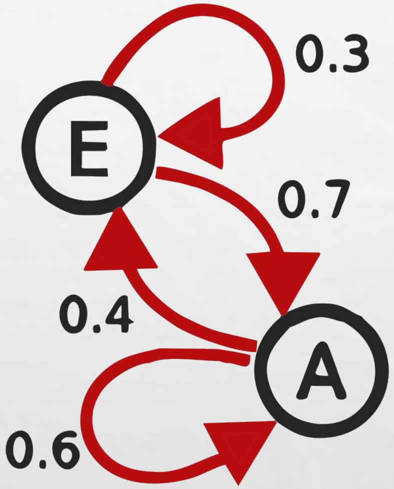
1.2 ANDREI ANDREYEVICH MARKOV

- MATEMÁTICO E PROFESSOR;
- FRAÇÕES CONTÍNUAS;
- CADEIA DE MARKOV.

1.3 QUANTUM CLOUD

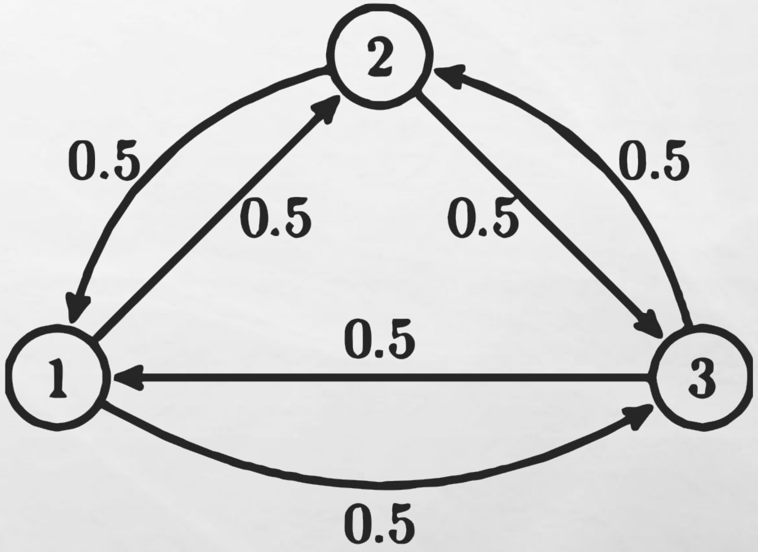
- ANTONY GORMLEY;
- PROJETADA POR UM COMPUTADOR;
- ALGORITMO DE CAMINHADA ALEATÓRIA.

2. INTRODUÇÃO



2.1 CADEIA DE MARKOV

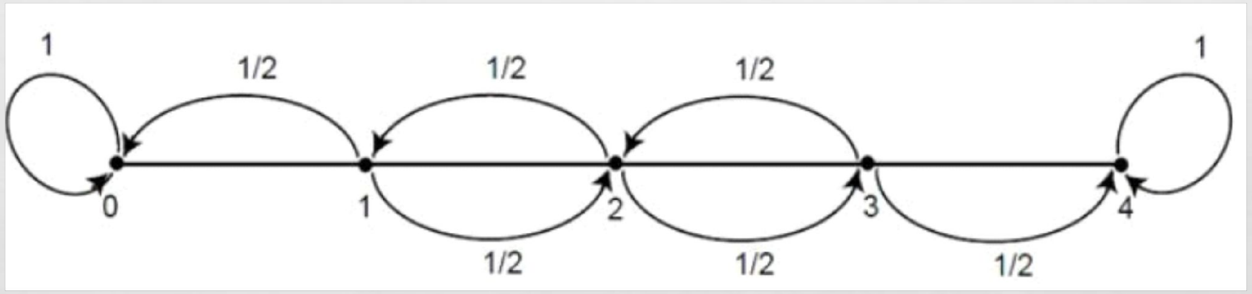
- ANDREI MARKOV;
- PROCESSO ESTOCÁSTICO;
- PROPRIEDADE MARKOVIANA.



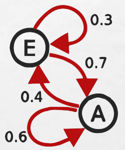
2.2 CASO ABSORVENTE

- UMA VEZ INSERIDO NO ESTADO, JAMAIS É DEIXADO ;

- EXEMPLO: UM BÊBADO CAMINHA NA RUA. CADA NÚMERO DE 1 A 3 REPRESENTA UM QUARTEIRÃO, ENQUANTO O NÚMERO 0 REPRESENTA A CASA E O NÚMERO 4 REPRESENTA O BAR.



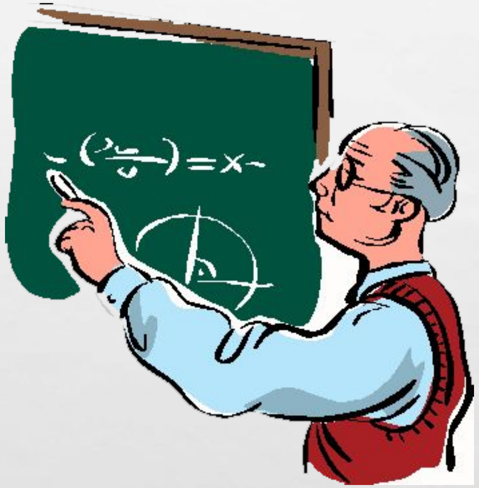
2.3 MODELANDO PROBLEMAS REAIS

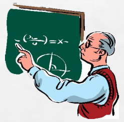


HÁ CADEIA DE MARKOV NA

- PSICOLOGIA;
- CLASSIFICAÇÃO DE PÁGINAS NA WEB;
- BOLSA DE VALORES;
- ETC.

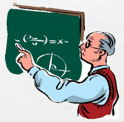
3. FUNDAMENTAÇÃO TEÓRICA





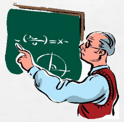
3.1 CLASSIFICAÇÃO DE ESTADOS

- ESTADO TRANSIENTE
- ESTADO ABSORVENTE



3.1.1 ESTADO TRANSIENTE

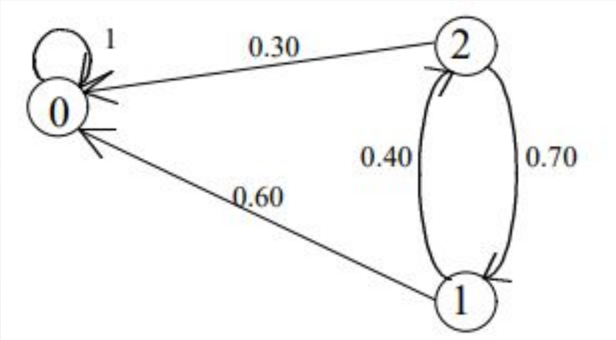
- POSSIBILIDADE DE JAMAIS RETORNAR AO ESTADO NOVAMENTE;
- S_I TRANSIENTE SE, E SOMENTE SE, EXISTIR UM S_J ($S_I \neq S_J$) QUE SEJA ACESSÍVEL A S_I.

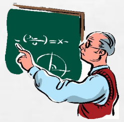


3.1.2 ESTADO ABSORVENTE

- ENTRANDO, JAMAIS IRÁ DEIXAR O ESTADO;
- i É ABSORVENTE SE, E SOMENTE SE, $P_{ii} = 1$.

3.1 CLASSIFICAÇÃO DE ESTADOS





3.2 FORMA CANÔNICA

- **MATRIZ DE TRANSIÇÃO**

$$\mathbf{P} = \begin{array}{cc} & \begin{array}{cc} Trans & Absor \end{array} \\ \begin{array}{c} Trans \\ Absor \end{array} & \left[\begin{array}{c|c} \mathbf{Q} & \mathbf{R} \\ \hline \mathbf{0} & \mathbf{I_r} \end{array}\right] \end{array}$$

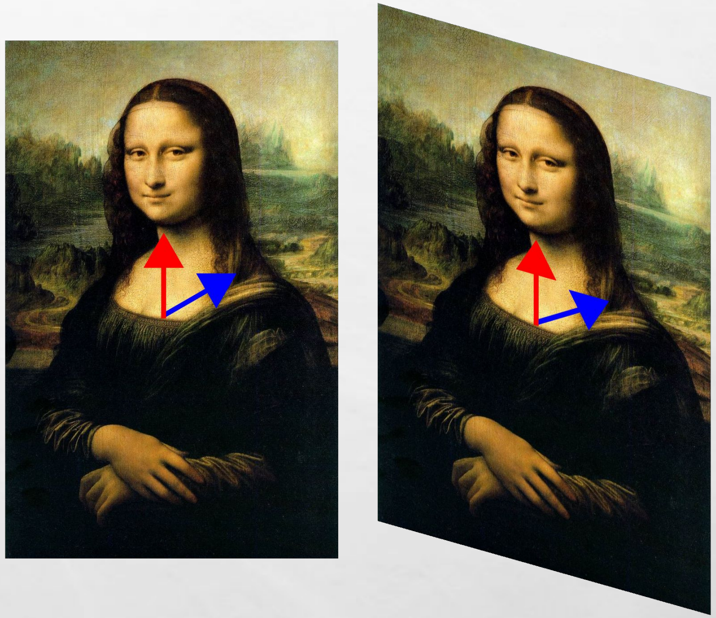
EM UMA CADEIA DE MARKOV ABSORVENTE, A PROBABILIDADE DE QUE O PROCESSO SEJA ABSORVIDO É IGUAL A 1.

3.3 MATRIZ FUNDAMENTAL

- $(I_{ij} - Q)^{-1}$
- VETOR t
- MATRIZ B

3.4 AUTOVALORES E AUTOVETORES

$$Ax = \lambda x$$



3.5 DECOMPOSIÇÃO ESPECTRAL

A = (S∧S-¹)

$$\begin{pmatrix} -1 & 0 & -2 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & 2 \\ 1 & 1 & 1 \\ -1 & 0 & -1 \end{pmatrix}$$

4. METODOLOGIA E RESULTADOS

4.1 MÓDULOS USADOS

```
import numpy as np
import matplotlib.pyplot as plt
import random
```

4.2 MATRIZ DE TRANSIÇÃO

```
def Gera_MatrizTransicao(direito, dimensao):
    esquerdo = 1 - direito
    MatrizTransicao = np.diag(esquerdo*np.ones(dimensao), k=-1) + np.diag(direito*np.ones(dimensao), k=1)
    MatrizTransicao[0, :] = 0
    MatrizTransicao[0, 0] = 1
    MatrizTransicao[dimensao, :] = 0
    MatrizTransicao[dimensao, dimensao] = 1
    return MatrizTransicao
```

```
[[1. , 0. , 0. , 0. , 0. , 0. , 0. , 0. ],
 [0.5, 0. , 0.5, 0. , 0. , 0. , 0. , 0. ],
 [0. , 0.5, 0. , 0.5, 0. , 0. , 0. , 0. ],
 [0. , 0. , 0.5, 0. , 0.5, 0. , 0. , 0. ],
 [0. , 0. , 0. , 0.5, 0. , 0.5, 0. , 0. ],
 [0. , 0. , 0. , 0. , 0.5, 0. , 0.5, 0. ],
 [0. , 0. , 0. , 0. , 0. , 0.5, 0. , 0.5],
 [0. , 0. , 0. , 0. , 0. , 0. , 0. , 1. ]]
```

4.3 FORMA CANÔNICA

```
A = [0, Y]
B = range(1, Y)
P = Gera_MatrizTransicao(X,Y)
Matriz_Transiente = P[np.ix_(B, B)]
Matriz_Absorvente = P[np.ix_(B, A)]
Matriz_nula = np.zeros([len(Matriz_Identidade),len(Matriz_Transiente)])
Matriz_Identidade = P[np.ix_(A, A)]
```

4.3 FORMA CANÔNICA

$$\begin{pmatrix}
0. & 0.5 & 0. & 0. & 0. & 0. & 0.5 & 0. \\
0.5 & 0. & 0.5 & 0. & 0. & 0. & 0. & 0. \\
0. & 0.5 & 0. & 0.5 & 0. & 0. & 0. & 0. \\
0. & 0. & 0.5 & 0. & 0.5 & 0. & 0. & 0. \\
0. & 0. & 0. & 0.5 & 0. & 0.5 & 0. & 0. \\
0. & 0. & 0. & 0. & 0.5 & 0. & 0. & 0.5 \\
0. & 0. & 0. & 0. & 0. & 0. & 1. & 0. \\
0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.
\end{pmatrix}$$

4.4 MATRIZ FUNDAMENTAL

```
Matriz_indent_fundamental= (np.eye(len(Matriz_Transiente)))
Subtracao = np.subtract(Matriz_indent_fundamental,Matriz_Transiente)
matriz_fundamental = np.linalg.inv(Subtracao)
np.set_printoptions(suppress = True, precision=3)
```

```
[[1.714 1.429 1.143 0.857 0.571 0.286]
 [1.429 2.857 2.286 1.714 1.143 0.571]
 [1.143 2.286 3.429 2.571 1.714 0.857]
 [0.857 1.714 2.571 3.429 2.286 1.143]
 [0.571 1.143 1.714 2.286 2.857 1.429]
 [0.286 0.571 0.857 1.143 1.429 1.714]]
```

4.4 MATRIZ FUNDAMENTAL

- **VETOR t**

```
Vetor_t = matriz_fundamental.sum(axis=1)
```

```
[ 6. 10. 12. 12. 10.  6.]
```

4.4 MATRIZ FUNDAMENTAL

- **MATRIZ B**

```
matriz_b = np.dot(matriz_fundamental,Matriz_Absorvente)
```

```
[[0.857 0.143]
 [0.714 0.286]
 [0.571 0.429]
 [0.429 0.571]
 [0.286 0.714]
 [0.143 0.857]]
```

4.5 DECOMPOSIÇÃO ESPECTRAL

- **AUTOVETOR**

```
(matriz S)

[[ 1.      0.      0.061 -0.127  0.605  0.437 -0.204 -0.303]
 [ 0.      0.     -0.231  0.411 -0.12  -0.329  0.499  0.471]
 [ 0.      0.      0.416 -0.513 -0.216 -0.41  -0.222  0.21 ]
 [ 0.      0.     -0.519  0.228 -0.269 -0.182 -0.4   -0.378]
 [ 0.      0.      0.519  0.228 -0.269  0.182  0.4   -0.378]
 [ 0.      0.     -0.416 -0.513 -0.216  0.41   0.222  0.21 ]
 [ 0.      0.      0.231  0.411 -0.12   0.329 -0.499  0.471]
 [ 0.      1.     -0.061 -0.127  0.605 -0.437  0.204 -0.303]]
```

4.5 DECOMPOSIÇÃO ESPECTRAL

- **AUTOVALOR**

```
np.set_printoptions(suppress= True,precision=3)
autovalor, autovetor = np.linalg.eig(Matriz_trans.T)
```

```
[ 1.      1.     -0.901 -0.623  0.901  0.623 -0.223  0.223]
```

4.5 DECOMPOSIÇÃO ESPECTRAL

- DIAGONALIZAÇÃO DE AUTOVALORES

```
matrizdiagonal = np.diag(autovalor)
```

```
Matriz Diagonal (matriz A)

[[ 1.      0.      0.      0.      0.      0.      0.      0.    ]
 [ 0.      1.      0.      0.      0.      0.      0.      0.    ]
 [ 0.      0.     -0.901   0.      0.      0.      0.      0.    ]
 [ 0.      0.      0.     -0.623   0.      0.      0.      0.    ]
 [ 0.      0.      0.      0.      0.901   0.      0.      0.    ]
 [ 0.      0.      0.      0.      0.      0.623   0.      0.    ]
 [ 0.      0.      0.      0.      0.      0.     -0.223   0.    ]
 [ 0.      0.      0.      0.      0.      0.      0.      0.223]]
```

4.5 DECOMPOSIÇÃO ESPECTRAL

- MATRIZ DE AUTOVETORES INVERSA (S^{-1})

```
matrizAutovetoresInversa = np.linalg.inv(autovetor)
```

```
[[ 1.     0.857  0.714  0.571  0.429  0.286  0.143  0.   ]
 [ 0.     0.143  0.286  0.429  0.571  0.714  0.857  1.   ]
 [ 0.    -0.233  0.419 -0.523  0.523 -0.419  0.233  0.   ]
 [-0.     0.425 -0.53   0.236  0.236 -0.53   0.425 -0.   ]
 [-0.    -0.449 -0.808 -1.008 -1.008 -0.808 -0.449 -0.   ]
 [ 0.    -0.531 -0.662 -0.295  0.295  0.662  0.531  0.   ]
 [-0.     0.544 -0.242 -0.436  0.436  0.242 -0.544 -0.   ]
 [ 0.     0.577  0.257 -0.462 -0.462  0.257  0.577  0.   ]]
```

4.5 DECOMPOSIÇÃO ESPECTRAL

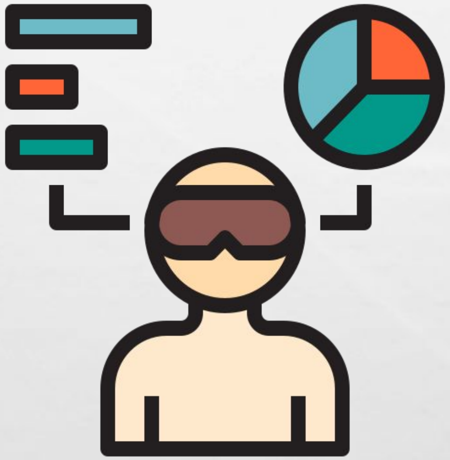
- $S\Lambda S^{-1}$

```
decomposição_sas = np.matmul(np.matmul(autovetor,matrizdiagonal),np.linalg.inv(autovetor))
```

```
Decomposição Espectral (SΛS-¹)

[[ 1.   0.5 -0.  -0.   0.   0.  -0.   0. ]
 [ 0.   0.   0.5  0.   0.   0.   0.   0. ]
 [ 0.   0.5  0.   0.5  0.  -0.  -0.   0. ]
 [ 0.   0.   0.5  0.   0.5  0.   0.   0. ]
 [ 0.  -0.   0.   0.5  0.   0.5  0.   0. ]
 [ 0.   0.   0.   0.   0.5  0.   0.5  0. ]
 [ 0.  -0.  -0.  -0.   0.   0.5  0.   0. ]
 [ 0.   0.   0.   0.  -0.  -0.   0.5  1. ]]
```

5. SIMULAÇÃO



OBRIGADO!

SE BEBER NÃO DIRIJA ☺