

Visualization of DNA sequences according to Chaos Game Representation (CGR)

24AIM144 Introduction to Data Compression

Team Members:

1. A. Guru Jaya Surya Yadav - CB.AI.U4AIM24101.
2. J. Tej Krishna Sai - CB.AI.U4AIM24117.
3. P. Teja Prakash Royal - CB.AI.U4AIM24136.
4. S. Ankith - CB.AI.U4AIM24147.

1. Abstract

The rapid expansion of DNA sequence data necessitates the effective visualization and analysis of data. Chaos Game Representation (CGR) is a new way of graphically representing nucleotide sequences in a compact format. In this work, a Python script is introduced for translating DNA sequences to CGR images such that researchers and bioinformaticians can uncover concealed patterns and pathologies in genomic information. The script tries to provide an interactive interface and sequence upload capability for the generation of CGR plots.

2. Introduction

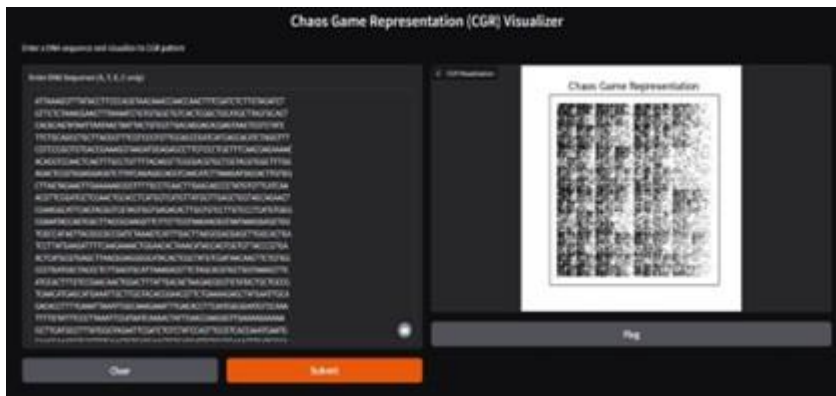
DNA sequences are four-base long sequences of Adenine (A), Thymine (T), Cytosine (C), and Guanine (G). Sequence analysis may provide important information about genetic characteristics, mutations, and illnesses. Conventional analysis is text-based and requires much time. Chaos Game Representation (CGR), found by Jeffrey in 1990, projects sequences on 2D space, and it forms pictures illustrating frequency and distribution of k-mers in a sequence. The graphical method is effective in recognizing patterns, repetition, or randomness in sequences in a matter of a few minutes.

3. Problem Statement

It is hard by text alone to comprehend and decipher long DNA sequences. There are insufficient graphical and intuitive tools to examine the pattern and distribution of the nucleotides of genomic sequences.

4. Objective

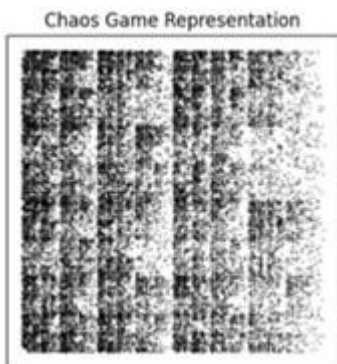
Create a basic, user-friendly web-based application or Gradio app for the visualization of CGR.



Accept raw DNA sequences as input. Accept raw DNA sequences as input.

```
TGTCTGGTAAAGGCCAACAACAA
CAAG
GCCAACTGTCACTAAGAAATCT
GCTGCT
GAGGCTTCTAAGAAGCCTCGGCA
AAAAC GTA
CTGCCACTAAAGCATACAATGTAA
CACAA
GCTTTCGGCAGACGTGGTCCAG
AACAAA CCC
AAGGAAATTTTGGGGACCAGGAA
CTAAT
CAGACAAGGAACTGATTACAAAC
ATTGG CCGC
AAATTGCACAATTTGCCCCCAGC
GCTTC
AGCGTTCTTCGGAATGTCGCGCA
TTGGC ATGG
```

Produce high-definition CGR shots of the given sequences.



Facilitate visualization of k-mer frequencies in genomic data

5. Methodology

Input Handling:Accept DNA sequencesas input in FASTA or string format.

CGR Algorithm:

1. Every vertexof a unit square is labeled with a nucleotide (A, T, C, G).
2. Start in the middle and half-step through the series,half-stepping to the appropriate corner based on the current nucleotide.
3. Plot all the points to constructthe CGR image.
4. Visualization: Plot and color the pointsusing matplotlib and seaborn in Python.
5. Web App: Frontendusing Gradio for simplicity.

6. Tools and Technologies

- Python
- Matplotlib / Seaborn
- Gradio
- NumPy

7. Results

1. Successful CGR plots generated for different DNA sequences.
2. Visualizations establishunmistakable trends by sequence lengthand makeup.
3. Tool allows researchers to explore sequencestructure intuitively.

8.Applications

- Bioinformatics research
- Genomic pattern recognition
- Mutation and anomalydetection
- Educational presentations of DNA structure

9.Conclusion

Chaos Game Representation is a powerful method for representing DNA sequences. The computer-generated representation bridges the gap between sequence information and intuitive sense through images.It gives scientists a new way of understanding genetic information.

10.Future Work

- Permit color-coding of higher-order k-mers.
- Embark on sequencealignment and clustering.
- Provide protein sequenceand codon mappingassistance.
- Add saving and exporting of CGR images.

11. References

1. Jeffrey H. J., Chaosgame representation of gene structure. *Nucleic Acids Res.* 18:2163–2170, 1990.
2. Deschavanne P. J., Giron A., Vilain J., Fagot G., Fertil B: Genomic signature: characterization and classification of species assessed by chaos game representation of sequences. *Mol. Biol. Evol.*, 16:1391-1399, 1999.
3. Kathleen A. Hill, Nicholas J. Schisler and Shiva M. Singh, ChaosGame Representation of coding regions of human globin genes and alcohol dehydrogenase genes of phylogenetically divergent species. *J Mol Evol.* 35:261-269, 1992.
4. Chitra Dutta and Jyotirmoy Das, Mathematical Characterization of Chaos Game Representation New Algorithms for Nucleotide Sequence Analysis. *J Mol Biol.* 228:715-719, 1992.
5. Goldman N., Nucleotide, dinucleotide and trinucleotide frequencies explain patterns observed in chaosgame representations of DNA sequences. *Nucleic Acids Res*, 21:2487-2491, 1993.
6. Almeida, J.S, Analysis of genomic sequences by chaos game representation. *Bioinformatics* 17: 429-437, et al. 2001.
7. Jijoy Josephan and Roschen Sasikumar, Chaos Game Representation for comparison of whole genomes. *BMC Bioinformatics* 7:243, 2006.
8. Deschavanne P. J., Giron A., Vilain J., Dufraigne C. and Fertil.

SourceCode:

```
import gradio as gr

import matplotlib.pyplot as plt

def generate_cgr(sequence):

    # Ensure uppercase and filter only A, T, G, C
    sequence = "".join([base for base in sequence.upper() if base in "ATGC"]) if len(sequence) < 2:
    return "Sequence too short!"

    # Assign corners to each base
    corners = {'A': (0, 0), 'T': (0, 1), 'G': (1, 1), 'C': (1, 0)}

    # Start from center x, y = 0.5, 0.5
    x_vals, y_vals = [x], [y]

    for base in sequence:
        cx, cy = corners[base]
        x = (x + cx) / 2
        y = (y + cy) / 2
        x_vals.append(x)
        y_vals.append(y)

    # Plot CGR
    fig, ax = plt.subplots(figsize=(4, 4))
    ax.plot(x_vals, y_vals, 'k.', markersize=0.5)
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_title("Chaos Game Representation")
    ax.set_aspect('equal')

    return fig

# Gradio interface
iface = gr.Interface(
```

```
fn=generate_cgr,  
inputs=gr.Textbox(label="Enter DNA Sequence (A, T, G, Only)",  
  
outputs=gr.Plot(label="CGR Visualization"), title="Chaos Game Representation (CGR) Visualizer",  
description="Enter a DNA sequence and visualize its CGR pattern"  
)  
iface.launch()
```