Segmentation of Brain Tumor Regions in MRI Images Using Graph Laplacian–Based Spectral Clustering

Ahmad Zaky Robbani - 13524045
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524045@std.stei.itb.ac.id, [2]ahmadzakyrobbani@gmail.com

***Abstract*—Magnetic Resonance Imaging (MRI) is the primary non-invasive method to visualize internal body condition such as brain tumors. However, automating tumor detection is a difficult area of research due to tumors not forming well-defined objects with their unclear boundaries and appearance like healthy tissues. This paper attempts to view effectiveness of spectral clustering in image segmentation of brain tumors.**

***Keywords*—Spectral Clustering, Magnetic Resonance Imaging, Brain Tumor, Image Segmentation.**

I. Introduction

Magnetic Resonance Imaging (MRI) provides high-contrast detailed visualization of tissues with non-invasive method. Precise segmentation of tumors from MRI is critical because treatment relies on information of tumor composition. Normally, multiple MRI modalities (T1, T1c, T2, FLAIR) are acquired to capture the biological characteristics of brain tissue. Each modality reveals complementary information that will be used to diagnose tumor composition. Due to the importance of tumor boundary segmentation, demand for automated or semi-automated methods of diagnosis is high. Despite that, the complex nature of tumor makes this field remain a difficult area of research.

Spectral clustering is an unsupervised image segmentation technique that has been generally used to analyze images. Spectral clustering represents image as graph where each node is a pixel and each edge is a similarity connection between two pixels. This changes the problem to a graph partitioning problem. Spectral clustering has succeeded in natural image segmentation where objects generally have distinct colors and textures. This makes the method an interesting candidate to solve brain tumor detection problems. In this paper, spectral clustering will be tested against the complex problem of MRI image processing.

II. Literature Review

A. Vectors

Vectors are fundamental mathematical objects used to represent data, which often has a magnitude and direction, and are represented as ordered list of numbers. Vectors may exist in any number of positive dimensions. In linear algebra, vectors must have these characteristics:

1. Vector addition
   Addition of two vectors in the same dimension results in a vector in that same dimension.
2. Scalar multiplication
   A vector multiplied by some scalar (number without direction) results in a vector in the same dimension

B. Matrix

Matrices (plural of matrix) are a way to organize and manipulate data represented as vectors. Matrices can be interpreted as linear transformation that transforms a vector and outputs a vector. Linear algebra generally revolves around these matrix concepts:

1. Matrix transpose
2. Matrix addition
3. Matrix multiplication
4. Inverse of a matrix

C. Graph

Graph is a structure composed of a set of nodes and a set of edges connecting two nodes. A graph can be represented by adjacency matrix. In an undirected simple graph, the adjacency matrix is a symmetrical matrix.

D. Degree Matrix and Graph Laplacian

Degree matrix is a matrix where the diagonal element is the sum of degree in the corresponding row of adjacency matrix. A graph Laplacian is the degree matrix subtracted by the adjacency matrix. The graph Laplacian captures the feature of graph connectivity.

E. Eigenvalue and Eigenvector

F. Spectral Clustering

G. K-Means Clustering

III. IMPLEMENTATION

A. Pipeline

B. Python Program

```python
path = "./data/BraTS2020_training_data/content/data/"

def load_slice(volume: int, slice: int):

    fullPath = path + "volume_" + str(volume) + "_slice_" + str(slice) + ".h5"
    with h5py.File(fullPath, "r") as f:
        image = f["image"][:].astype(np.float64)
        flair = image[:, :, 3]

        return flair
```

```python
def pixel_index(x, y, width):
    return y*width + x

def get_neighbors(x, y, height, width):
    neighbors = []

    directions = [
        (-1, -1), (0, -1), (1, -1),
        (-1,  0),          (1,  0),
        (-1,  1), (0,  1), (1,  1)
    ]

    for dx, dy in directions:
        nx, ny = x + dx, y + dy
        if 0 <= nx < width and 0 <= ny < height:
            neighbors.append((nx, ny))

    return neighbors
```

```python
def build_adjacency(image):
    height, width = image.shape
    n = height * width

    rows, cols, data = [], [], []

    for y in range(height):
        for x in range(width):
            index = pixel_index(x, y, width)
            neighbors = get_neighbors(x, y, height, width)
            for (xNeighbor, yNeighbor) in neighbors:
                indexNeighbor = pixel_index(xNeighbor, yNeighbor, width)

                if indexNeighbor > index:
                    dist2 = (x - xNeighbor)**2 + (y - yNeighbor)**2
                    weight = exp(-(image[y, x] - image[yNeighbor, xNeighbor])**2 / (2* 0.5**2) - dist2/(2))
                    rows.extend([index, indexNeighbor])
                    cols.extend([indexNeighbor, index])
                    data.extend([weight, weight])

    W = sparse.csr_matrix((data, (rows, cols)), shape=(n, n))
    return W
```

```python
def compute_normalized_laplacian(W):
    degrees = np.array(W.sum(axis=1)).flatten()
    degrees_inv_sqrt = np.zeros_like(degrees)
    nonzero = degrees > 0
    degrees_inv_sqrt[nonzero] = 1.0 / np.sqrt(degrees[nonzero])

    D_inv_sqrt = sparse.diags(degrees_inv_sqrt)

    L_norm = sparse.eye(W.shape[0]) - D_inv_sqrt @ W @ D_inv_sqrt
    return L_norm
```

```python
def spectral_embedding(W, k, epsilon=1e-8):
    from scipy.sparse.linalg import eigsh, lobpcg

    L_norm = compute_normalized_laplacian(W)
    n = L_norm.shape[0]

    num_eigs = min(n - 1, max(k + 3, k + 10))

    try:
        evals, evecs = eigsh(L_norm, k=num_eigs, which='SA', tol=1e-3, maxiter=2000)
    except Exception:
        rng = np.random.default_rng(42)
        X0 = rng.standard_normal((n, num_eigs))
        evals, evecs = lobpcg(L_norm, X0, tol=1e-3, maxiter=500)

    order = np.argsort(evals)
    evals = evals[order]
    evecs = evecs[:, order]

    nonzero_mask = evals > epsilon
    if np.sum(nonzero_mask) < k:
        raise ValueError(f"Requested {k} nonzero eigenvectors, but only {np.sum(nonzero_mask)} available.")

    U = evecs[:, nonzero_mask][:, :k]
    return U
```

```python
def cluster(U):
    U = U / np.linalg.norm(U, axis=1, keepdims=True)
    kmeans = KMeans(n_clusters=2, random_state=42, n_init=10)
    labels = kmeans.fit_predict(U)

    return labels
```

```
image = load_slice(5, 77)

W = build_adjacency(image)

U = spectral_embedding(W, k=2)

labels = cluster(U)

height, width = image.shape
mask_2d = labels.reshape(height, width)

plt.imshow(mask_2d, cmap='gray')
plt.show()
```

V. Conclusion

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

VI. Appendix

Appendixes, if needed, appear before the acknowledgment.

VII. Acknowledgment

The preferred spelling of the word “acknowledgment” in American English is without an “e” after the “g.” Use the singular heading even if you have many acknowledgments. Avoid expressions such as “One of us (S.B.A.) would like to thank ... .” Instead, write “F. A. Author thanks ... .” Sponsor and financial support acknowledgments are placed in the unnumbered footnote on the first page.

References

[1] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, et al. "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)", IEEE Transactions on Medical Imaging 34(10), 1993-2024 (2015) DOI: 10.1109/TMI.2014.2377694W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[2] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J.S. Kirby, et al., "Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features", Nature Scientific Data, 4:170117 (2017) DOI: 10.1038/sdata.2017.117B. Smith, “An approach to graphs of linear forms (Unpublished work style),” unpublished.

[3] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, et al., "Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction in the BRATS Challenge", arXiv preprint arXiv:1811.02629 (2018)

[4] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, et al., "Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-GBM collection", The Cancer Imaging Archive, 2017. DOI: 10.7937/K9/TCIA.2017.KLXWJJ1Q

[5] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. Kirby, et al., "Segmentation Labels and Radiomic Features for the Pre-operative Scans of the TCGA-LGG collection", The Cancer Imaging Archive, 2017. DOI: 10.7937/K9/TCIA.2017.GJQ7R0EF.

[6]

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

Ahmad Zaky Robbani - 13524045