MARCH 1, 2024

ASSIGNMENT #1

BDA - LSH

HARRIS HASSAN, ABDULLAH NADEEM & FADIL AWAN
DS (D)

Part 1:
Q1:

Overall, the code implements Locality Sensitive Hashing for approximate nearest neighbor search on the CIFAR-10 dataset. It utilizes LSH to efficiently retrieve nearest neighbors for query images based on their features, facilitating fast similarity search in high-dimensional feature space. Following are the five techniques/functions that are used to perform LSH.

Data Loading and Preprocessing:

- Loads the CIFAR-10 dataset stored in the specified directory. It loads images and their corresponding labels from the dataset directory.
- Preprocesses the loaded images by resizing them, converting to grayscale, computing histograms, performing edge detection, and extracting features. It returns a feature matrix where each row represents the features of an image.

Visualization:

- Visualizes the preprocessed images, grayscale images, edges, and color histograms for the first five images in the dataset.
- CV2 library used for grayscale imaging and canny edge detection filter used as it is the best filter for detecting edges.
- Color histograms for red, green and blue also displayed.

LSH Class:

- **Init:** Initializes the LSH parameters including the number of hash tables (**number of tables**), the number of hash functions per table (**number of functions**), the number of buckets per table (**number of buckets**), and the dimensionality of the feature vectors (**number of dimensions**). It also initializes the projection matrices for each table and the hash tables themselves.

- **Projection matrix**: Generates a random projection matrix for each table. These matrices are used to project high-dimensional feature vectors onto lower-dimensional space.

- **Hash function**: Computes the hash value of a vector using a given projection matrix. This hash value is used to index into the hash table.

- **Hash vector**: Hashes a given vector into all hash tables. It computes the hash value for each table and updates the corresponding hash table with the hashed vector.

- **Query**: Executes a query on the LSH data structure. It hashes the query vector into each table, retrieves the candidates from each table, and aggregates the results.

Helper Functions:

- Computes the Euclidean distance between two vectors.
- Computes the cosine similarity between two vectors.

Execution:

- **Loading and Preprocessing Data**: Loads the CIFAR-10 training dataset, preprocesses the images, and extracts features from them.

- **LSH Instantiation and Population**: Instantiates the LSH object with specified parameters and populates the LSH tables with the training data.

- **Query Execution**: Selects a subset of images from the training set as queries, executes queries against LSH tables, and retrieves nearest neighbors using both Euclidean distance and cosine similarity.

The implementation employs Locality Sensitive Hashing (LSH), a technique adept at approximate nearest neighbor search in high-dimensional spaces. LSH is particularly suited for scenarios like image retrieval, where traditional methods face scalability issues due to the curse of dimensionality. By dividing the feature space into hash buckets using multiple hash functions across tables, LSH efficiently indexes data, increasing the likelihood of similar data points being hashed into the same bucket. This approach drastically reduces the search space, enabling faster retrieval of approximate nearest neighbors while maintaining reasonable accuracy.

Furthermore, the choice of both Euclidean distance and cosine similarity metrics for evaluating nearest neighbors offers versatility in similarity measurement. Euclidean distance provides a straightforward measure of geometric distance between vectors, while cosine similarity is effective in capturing the directional relationship between vectors, irrespective of their magnitude. This combination allows for a comprehensive exploration of the similarity landscape, catering to diverse use cases and data characteristics prevalent in image datasets like CIFAR-10. Overall, this implementation leverages LSH's efficiency and the versatility of similarity metrics to facilitate effective image retrieval in high-dimensional spaces.

Part 2:
Q2:

Overview:

The provided code implements Locality Sensitive Hashing (LSH) for approximate nearest neighbor search on the CIFAR-10 dataset. The application integrates LSH with a Flask web application to provide a user-friendly interface for image retrieval and analysis. The implementation consists of several components, including data loading and preprocessing, LSH integration, Flask routes, HTML templates, and visualization.

Flask Application (app.py):

- Initialization: Initializes the Flask application and configures routes and settings.
- Routes: Defines routes for different functionalities, including the home page, form submission, and image upload.
- File Upload: Handles file uploads from the user interface and saves uploaded images to the server.
- LSH Integration: Integrates LSH functionality for image retrieval and similarity search.
- Query Execution: Executes LSH queries based on user input and displays results on the web interface.
- Error Handling: Implements error handling to manage exceptions and provide appropriate feedback to users.

HTML Templates:

- Index HTML (index.html): Provides the main interface for users to interact with the application. It allows users to upload images and initiate similarity search queries.
- Similar Images HTML (similar_images.html): Displays the results of similarity search queries. It shows the uploaded image and a list of similar images retrieved using LSH.
- Error HTML (error.html): Renders error messages to users in case of invalid inputs or other errors.

Execution:

- User Interaction: Users interact with the web interface by uploading images and initiating similarity search queries.
- Server Processing: The Flask server receives user requests, processes uploaded images, executes LSH queries, and generates responses.
- Results Display: The results of similarity search queries are displayed on the web interface using HTML templates, providing users with visual feedback on image similarity.

Conclusion:

The integration of Locality Sensitive Hashing (LSH) with a Flask web application facilitates efficient image retrieval and similarity search in the CIFAR-10 dataset. By combining LSH's scalability and fast query execution with a user-friendly interface, the application enables users to explore and analyze image data interactively. The seamless interaction between users and the application, coupled with the backend processing powered by LSH, demonstrates the innovative application of LSH in image processing and analysis.