Mining Massive Data (SS2025)

Programming Assignment 1

Locality Sensitive Hashing (LSH)

Availability date: 06.03.2025
Submission due date: 10.04.2025, 23:59
Peer-review due date: 17.04.2025

Number of tasks: 1
Maximum achievable points: 100 points (85 + 15 . . . Submission / Quality of your review)

General Remarks

- This is one of 3 programming assignments. For each assignment you can earn up to 100 points. Details on grading are provided below.

- The deadline for this assignment is noted on the header. No deadline extensions will be granted. Upload your solution by the deadline on the course's Moodle page [4]. If you fail to upload your solution by the deadline you will not be awarded any points.

- If you have questions or encounter problems, do not hesitate to contact us.

- Solve this programming assignment in Python. Don't use any non-standard libraries in your code, e.g., only use `numpy`, `scipy`, `scikit-learn`, unless explicitly mentioned.

- Make sure that your code is executable, e.g., if you created a jupyter notebook, make sure that the notebook can be executed without error when you reset the kernel and execute the notebook from the beginning. **If your code is not executable, we reserve the right to not award any points.**

- Solve the tasks on your own (considered group-wise). If we observe plagiarism or group work, we reserve the right to not award any points.

- Pack your code, your report, and execution instructions into a single .zip file with the following name: `group(group number).zip` file (e.g., `group01.zip`) and upload it on Moodle.

- Only **one** team-member submits the zip file in the Moodle system.

Questions

Please ask questions primarily on the respective Moodle forum. If you don't get a response in reasonable time (96 hours), get in touch with us via email (Timo Klein, Sebastian Tschiatschek).

Learning Objectives

The learning objective of this programming assignment is to to familiarize yourself with LSH. Do not use an existing LSH library.

Local Sensitive Hashing (LSH) for Item Search

Goal. The goal of this assignment is to implement an approximate nearest neighbor algorithm for classifying the genre of new music tracks based on efficiently identifying similar music tracks whose genres are known. You will base your implementation of the nearest neighbor search on local sensitive hashing (LSH) using the random projection method.

Data. As data source, use the Free Music Archive (FMA)[3]. For this exercise, we will use the tracks from the *medium* dataset, as explained in the corresponding paper [2] – you don't have to download the *medium* dataset but only the file `fma_metadata.zip`. The medium dataset contains 16 genres.

Data Preparation and Train/Validation/Test Splits. You can use pandas to load the relevant data files (`tracks.csv` and `features.csv`). In particular, for loading the tracks, you can use the command `df_tracks = pandas.read_csv('tracks.csv', index_col=0, header=[0, 1])` (you probably have to adjust the path to the tracks file). After this, you need to filter out the relevant tracks from `tracks.csv` corresponding to the *medium* dataset. You can do this by filtering the original list of tracks as follows: `df_tracks = df_tracks[df_tracks['set']['subset'] == 'medium']`.

Furthermore, you have to split your dataset into training, validation and test data. Use the predefined splitting as explained in the FMA paper, which can be found in the *set/split*-column for each track in `tracks.csv`. It ensures that the numbers of tracks per genre is balanced in the training, validation and test data. When loading the data as explained above using pandas, the genre information is available in the *track/genre_top*-column. The 8 genres you are considering are *Hip-Hop, Pop, Folk, Rock, Experimental, International, Electronic, and Instrumental.*

Assignment

Implement a local sensitive hashing algorithm using the random projection method applied on the extracted features in the `features.csv` data file (provided by the FMA). A description of LSH and the random projection method, can be found in this blog post[1] and this paper[2]. Based on your LSH implementation and the training data, implement an approximate nearest neighbour search to classify the genre of new music tracks.

Construction of the Random Matrix

Instead of drawing the elements r_{ij} for the random projection matrix $\mathbf{R}$ from a Gaussian distribution, use the following construction method introduced by Achlioptas [1]:

$$r_{ij} = \sqrt{3} \cdot \begin{cases} +1 & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ -1 & \text{with probability } \frac{1}{6} \end{cases}$$

General Training Algorithm and Hyperparameter Optimization

(1) For each track in the (standardized) training data, calculate its hash value of length l using LSH. Do so for n different hash tables and add the track to its corresponding bucket in each hash table.

(2) For each track t_i in the validation data, find similar music tracks using LSH. A music track is defined as similar if it is in the same bucket as t_i in one of the n hash tables.

(3) Calculate the actual similarity of t_i to all similar music tracks based on their feature vectors and some similarity metric m. As a genre prediction for t_i, use the majority genre of its k-nearest-neighbours defined as the k most similar music tracks to t_i as ranked by m found in the same bucket as t_i.
Note: Think about how to treat cases where there are less then k music tracks similar to t_i.

(4) Evaluate the classification accuracy of your algorithm on the validation data.

As possible similarity measures in step **(3)**, consider both the *cosine similarity* and the

[1]https://medium.com/data-science/locality-sensitive-hashing-for-music-search-f2f1940ace23
[2]https://cs-people.bu.edu/evimaria/cs565/kdd-rp.pdf

Euclidean distance.

The parameters l, n and k are free parameters of the algorithm and have to be set to sensible values through systematically experimenting with different choices and evaluating each choice by following the four steps of the general training algorithm outlined above. Test at least 3 different values for each of the parameters l, n, k and two choices for m.

Once you have found a set of parameters l, n and k and a similarity measure m which performs well on the verification data set, retrain your algorithm with these parameter choices using the training and validation data set as training data. Then evaluate the performance of your so trained algorithm on the test data and report your results. It is important to keep in mind, that you are only allowed to evaluate the performance of your algorithm on the test set **once** and you cannot change your parameter set choice or algorithm afterwords. If you would change your algorithm or set of parameters after evaluating the performance on the test set, this would correspond to using the test set as validation set and therefore as part of the training process. Then the performance of your algorithm on the test set would not representative of the true generalization error, i.e., a measure for how accurately an algorithm is able to predict the correct outcome on previously unseen data, anymore.

Report

Write a report about your work on this assignment, your findings and results. You have to write only **one report per group**. In particular, include the following information:

☐ Briefly discuss your implementation of LSH and of the approximate nearest neighbour algorithm.

☐ Detail how you trained your algorithm and how you performed the hyperparameter optimization. Report tested parameter and the results for these parameters.

☐ Detail why you settled on a specific choice of l (hash length), n (number of hash tables), k (number of nearest neighbours for the prediction) and similarity measure m.

☐ Report the classification accuracy of your algorithm on the test set.

☐ Comment on why the chosen random projection method could be beneficial to drawing r_{ij} from a Gaussian distribution?

☐ Comment on how the runtime of your approximate nearest neighbor algorithm in comparison to that of an exact nearest neighbor search. (You can estimate runtime for the latter case.) If your implementation is slower than the exact search, speculate about possible reasons; comment on how these aspects might change when working with larger datasets.

☐ Report how you treat music tracks for which there are less than k other similar tracks.

☐ How much time did you spend on the assignment (including the writing of the report; please provide an average for all of your team members)? This will not be used for grading.

☐ Who did what? This will not be used for grading. This part is meant for you as a group to reflect on whether you shared the workload fairly and make adjustments for the next assignment if you observe imbalances.

References

[1] Dimitris Achlioptas. "Database-friendly random projections: Johnson-Lindenstrauss with binary coins". In: *J. Comput. Syst. Sci.* 66.4 (2003), pp. 671–687. DOI: 10.1016/S0022-0000(03)00025-4. URL: https://doi.org/10.1016/S0022-0000(03)00025-4.

[2] Michaël Defferrard et al. "FMA: A Dataset for Music Analysis". In: *18th International Society for Music Information Retrieval Conference (ISMIR)*. 2017. arXiv: 1612.01840. URL: https://arxiv.org/abs/1612.01840.

[3] Michaël Defferrard et al. *Free Music Archive*. https://github.com/mdeff/fma. 2017.

[4] *Mining Massive Data Moodle Page*. URL: hhttps://moodle.univie.ac.at/course/view.php?id=445688.