GEO5017 A2
Objects Classification from AirBorne LiDAR Data

1 Introduction

We live in a 3D world that is composed of urban entities, such as buildings, lampposts, mailboxes, and vegetation. During the last decades, there has been an ever-increasing demand, both by academia and industry, for analyzing and interpreting 3D urban geoinformation. A variety of research works have been devoted to the problem of 3D scene understanding. Successful interpretation of urban scenes allows for various modern applications such as urban modeling, autonomous driving, and environment monitoring.

Understanding the urban environment from 3D airborne LiDAR data has been extensively studied in recent years. Such data is vastly available in many countries. Compared to 2D imagery, point clouds provide more accurate 3D measurements with higher spatial resolution. Classification of point clouds is a fundamental task in 3D scene understanding, which aims to assign each point cloud a meaningful semantic label (e.g., building, tree, car). Semantic segmentation is a more complex task, assigning per point a label. In other words, segmentation can be viewed as dense classification on the point level.

In this assignment, instead of solving the semantic segmentation problem, we will focus on classification only. You will be working on classifying a set of pre-segmented individual urban objects into categories using supervised learning techniques (i.e., SVM, random forest).

2 The tasks

2.1 Feature engineering

The feature engineering contains the following two parts in this assignment:

- *Feature design.* You will need to design a set (≥ 6) of features for the task of classification. There are no standard features you can directly use. It could be helpful to visualize and observe the point clouds and their distribution of different types of objects to get some intuition. Be creative! Any quantities that can separate different types of objects can be good features and are worth looking into. You are also welcome to refer to the features introduced in previous works [1, 2, 3][1].

[1]Pay attention: many features in these works cannot be directly used in this assignment because they were designed for semantic segmentation, which is different from classification

- *Feature selection.* Use the metric(s)[2] introduced in our lecture to quantitatively analyze the effectiveness of your proposed features, and select the 4 most effective ones. Then what conclusion can be drawn from this feature selection process?

2.2 Classification

Choose a reasonable train-test split ratio (e.g., 6:4), carry out point cloud classification using the two classifiers introduced in the course, i.e., SVM and RF. In this assignment, we specifically ask you to use the implementation of the classifiers from Scikit-Learn. For the SVM classifier, please try different Kernel functions and then recommend the most promising one (and justify your choice in the report). For both classifiers (i.e., SVM, RF), try different combinations of hyper-parameters and find the best model with the highest performance.

3 Evaluation and analysis

Your evaluation and analysis (and report) should include:

- *Hyperparameters.* How do different key hyperparameters of a model affect its performance? Based on this analysis, explicitly make a recommendation of your final model for each classifier.
- *Learning curve.* From each recommended model of the classifiers (i.e., SVM, RF), gradually increase the train-test split ratio (e.g., 1:9, 2:8, ..., 9:1), evaluate the performance (e.g., error rate or OA) for each split, and make a plot of the performance with respect to the number of samples of the training set. This plot gives you the learning curve of your model. What can you observe from the curve?

  Note: your implementation must explicitly contain each of the above mentioned steps. Some machine learning libraries provide very convenient functions for generating and visualizing learning curves. For example, the learning_curve function in Scikit-Learn implements all these major steps in a single function. Using such functions is not allowed in this assignment (20% deduction otherwise).
- *Error analysis and comparison.* Provide an error analysis of your final model for both classifiers. Demonstrate a confusion matrix of your results. In what categories does your model do a good job? In what categories does your model make the most mistakes in prediction? Choose the most suitable metric for the evaluation and comparison, and don't forget to justify your choice of the metric. Based on this metric, which classifier has a better performance?

4 Dataset

We will use 500 point clouds of urban objects (The dataset is distributed with this assignment). For each point cloud, its ground truth label is encoded as follows (based on

[2]The optimal way of doing feature selection is to brute-force search for all possible combinations of features, evaluate their accuracy over the input dataset, and pick the feature combination that gives the highest performance. However, such a method is usually too computationally expensive, especially when the feature set and the dataset become large. Thus, in this assignment, you're explicitly asked to use the sub-optimal techniques for feature selection.

the base names of the point cloud files):
1) 000 - 099: building;
2) 100 - 199: car;
3) 200 - 299: fence;
4) 300 - 399: pole;
5) 400 - 499: tree.

5 Submission

Please compress all the following into a single archive titled **GEO5017_A1_Group_XX.zip** (where "XX" is your group ID. In case you have an updated version, please append "_V2" to its title) and submit it to BrightSpace:

- **Report** (max 3 pages excluding figures and tables) containing the following sections:
  - **Introduction**
    Motivation and goal of this assignment. (5%)
  - **Methodology**
    * Unambiguous definition of the initial features. (10%)
    * Description of the selection process for the 4 most effective features. (10%)
  - **Experiments and evaluation**
    * Analysis of key hyperparameters' effects and recommendation of final model for each classifier. All the subsequent tasks should be based on your final recommended models. (10%)
    * Generation and analysis of learning curves for both classifiers. (10%)
    * Description of classification results and error analysis for both classifiers. (10%)
    * Comparison of the results from the two classifiers. (5%)
  - **Conclusion**
    * Conclusion drawn from the assignment. (5%)
    * Suggestions for achieving better results. (5%)
  - **A short description of who did what**.

  To ensure formal scientific writing, the following rules apply for assessment:
  - Any misunderstanding or misconception of main concepts: 10% deduction.
  - Multiple unclear or ambiguous descriptions: 10% deduction.
  - Mulitple typos, grammar issues, format issues (e.g., a figure/table without a caption or multiple figures/tables with the exact same caption, unindexed or unreferenced figures/tables), consistency issues (e.g., upper case and lower case used interchangeably, normal font and italic font used interchangeably, misuse of symbols in notations): 10% deduction.
  - The report is more than half a page over length: 10% deduction.

- **Source code**
  - The source code, archived in a 'code' subfolder. The code should build, run, and reproduce your results without changes. (30%)
    - For Python code, **only** '*.py' files are accepted. If you use Jupyter Notebook for development, make sure you submit '*.py' files ('*.ipynb' files will not be accepted). For C++ code, **only** '*.cpp' files are accepted.
    - If you have multiple source code files that should run in a specific order, **do** name them in a way such that they are well ordered (e.g., '1_features.py', '2_train.py', '3_test.py', '4_visualization.py'). If no such information is provided, we will **expect a 'main.py' file** as the only entry point.
    - Provide a 'ReadMe.txt' file to briefly explain how to run the code and reproduce the results, e.g., dependence on external libraries/packages, the path to data, and where to find the results in case you save results or figures into files.
    - It is recommended to test your code on different operating systems, as your code may be evaluated on an OS different from that it was developed.
    - Please **do NOT include data** in your submission.
  - [*optional*] Provide a link to the GitHub repository (if you use GitHub) in the 'Experiment' section of your report. You are encouraged to collaborate with your teammates on GitHub.

Note: The report should be as concise as possible (within 3 pages excluding figures, tables, and references), but it should provide sufficient information to re-implement your methods and reproduce your results. Try to use the mathematical language (i.e., equations) as much as possible. You will have to omit less important details and discussions. Irrelevant descriptions and discussion may lead to the deduction of points.

References

[1] Chao-Hung Lin, Jyun-Yuan Chen, Po-Lin Su, and Chung-Hao Chen. Eigen-feature analysis of weighted covariance matrices for lidar point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 94:70–79, 2014.

[2] Hugues Thomas, François Goulette, Jean-Emmanuel Deschaud, Beatriz Marcotegui, and Yann LeGall. Semantic classification of 3d point clouds with multiscale spherical neighborhoods. In *2018 International conference on 3D vision (3DV)*, pages 390–398. IEEE, 2018.

[3] Weixiao Gao, Liangliang Nan, Bas Boom, and Hugo Ledoux. Sum: A benchmark dataset of semantic urban meshes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 179:108–120, 2021.