

An Expectation Maximization Algorithm for LiDAR Point Cloud Classification

Nguyen Thi Huu Phuong

Department of Information Technology, Hanoi University of Mining and Geology, Hanoi 11900, Vietnam

Abstract: LiDAR (Light Detection and Ranging) technology is now commonly used in geospatial technology when it is an active remote sensing technology and capable of collecting data on large areas. However, with a large dataset of measurement areas, selecting and using the data in accordance with the research purpose takes a lot of time to conduct the classification of points. The algorithm selection in LiDAR data processing problem is important in the use of lidar data. EM (Expectation Maximization) algorithm is a typical algorithm of data mining, with the advantage of being easy to install and understand the algorithm used in classification problems. In this study, the author improved the EM algorithm in classification of lidar point cloud by using scheduling parameters to reduce the convergence time of the algorithm.

Key words: LiDAR point cloud, EM, scheduling parameter.

1. Introduction

Since its inception in the 70s and 80s of the last century, LiDAR (Light Detection and Ranging) technology has been increasingly applied widely in many areas of society. Areas of application of LiDAR technology are such as: topographic survey and mapping; forestry which mainly uses LiDAR to assess and make statistic on timber production, correlation of factors such as canopy, canopy thickness etc.; flooding mapping; coastal applications such as coastal erosion management and prediction, coastal flood monitoring and forecasting; seabed topography; traffic mapping; mobile phone network, etc.

In recent times, studies on LiDAR technology, integrating LiDAR with traditional systems and applying LiDAR technology in various fields of social life have been published more and more. That proves, LiDAR is asserting its superiority compared to other earth observation technologies such as radar, satellites, In particular, studies on the use of EM (Expectation Maximization) algorithm in LiDAR point cloud classification were published as in 2015,

the authors Kun Zhang, et al. applied Kmeans to classifying point cloud by point density. The authors [1] used the density of point cloud as a condition to select the number of clusters to conduct classification and as a condition for convergence.

Chang, et al. [2] proposed the method of automatic extraction of terrain data from LiDAR data. By using a new algorithm that has the right plane combination and statistical filtering to eliminate hypothetical errors, ground and non-ground points will be divided. The algorithm has been tested by authors [2] using simulations and real data, and is proven with complex data in urban areas. In their study, Rodriguez, et al. [3] used point detection algorithm to detect and classify urban objects and trees from 3D MLS (Mobile Laser Scanning) data and TLS (Thin Plate Spline). The method includes automatic segmentation to eliminate parts unrelated to the segmentation process along the object. These objects are segmented using the RX (Reed and Xiaoli) algorithm and then clustered to divide the object layer into layers such as trees, artificial lakes, etc. The algorithm was tested with two-point clouds that were captured by two different sensors. In the test area, the authors [3] found that with the algorithm of detecting anomalies, the results

Corresponding author: Nguyen Thi Huu Phuong, master, research field: geoinformatics, software engineering.

were 95% accurate with two classes classified as trees and artificial structures such as lakes.

The same idea of LiDAR point cloud classification using point classification method, the authors Keng, et al. [4] had their own results. To make the classification process, the authors used 3D point cloud, this point cloud will be segmented into independent segments, then some features of the object are calculated and finally the points will be automatically categorized according to these characteristics. The test data set used in the study was taken from Kung Cheng National University. By providing the error matrix, the author assessed the accuracy of the method, the overall accuracy and the Kappa index indicating the compatibility of classification results when using OBC (Object-Based Classification) and human identity, spatial information such as the characteristics of points provided by point clouds that can be provided is diverse, abundant and powerful as the basis of classification [4]. LiDAR data are a relatively large data set and have a wide coverage across the scanning area, so sorting as a job is not easy. The point filtering process is key for almost every application with the LiDAR point cloud. With point filtering algorithms, parameter settings and increased convergence thresholds increase the accuracy of the point filtering process. Based on that idea, Hui, et al. [5] proposed the algorithm of non-threshold filtering based on the expected maximization algorithm (EM), classify the point cloud into two groups of ground and non-ground. EM is used to estimate the maximum expected for the GMM (Gaussian Mixture Model) parameter. After a number of iterations, the points of the point cloud are labeled, and the authors found that the proposed algorithm worked well for points belonging to the non-ground group, with only 4.48% of errors smaller than 8 algorithms. Point filtering math is reported in ISPRS (International Society for Photogrammetry and Remote Sensing) [5]. In his doctoral dissertation, Maligo [6] proposed a two-layer classification model,

the first layer was built on the GMM model and the second consisted of iterating an intermediate class group to select the grade correct for initial classification purpose. In particular, the GMM model is identified in the training class of unsupervised classification and defines a set of intermediate classes. Test results show that the proposed system works well on two data sets with an accuracy of 0.8-0.89 [6]. The creation of 3D models of urban areas is currently of interest to scientists. For a data set with a high-density point, containing a lot of noise, choosing an appropriate algorithm is absolutely necessary. Zhu, et al. [7] have proposed a point cloud classification algorithm based on multi-level semantic relationships. The input of the algorithm is a point cloud through transformations based on the uniformity of the point and the binding of neighboring points, the points are classified into layers with an accuracy of 93.55%. However, the proposed algorithm is still limited with received noise [7]. Lodha, et al. [8] used the algorithm to maximize expectations in classifying LiDAR ground scattering data into four groups: roads, grass, tall buildings and trees. To carry out the classification problem, the authors used five characteristics of LiDAR data: height, height variation, normal variation, LiDAR return intensity and image intensity. Accuracy of 94% was obtained for the 8 square mile area. Based on the selected parameters and models, the EM algorithm is appropriate for the classification area [8]. Meanwhile, Hui, et al. [9] showed that the EM algorithm fully meets the requirements of the automatic DTM (Digital Terrain Model) extraction problem. The error of the proposed algorithm is 16.78% lower than the traditional PTD (Progressive Triangulated irregular network Densification) algorithm and reduces the system error to 31.95% [9].

LiDAR data classification using the CRF (Conditional Random Field) algorithm of Luo and Sohn [10] gave an approach to use contextual information as an input to the classification problem. The classification problem is expressed using the

GMM model, the parameters of the model are selected by optimizing the iteration with the expected maximization algorithm. The CRF algorithm will be used to analyze the contextual information of the data. Through testing, the authors have shown that the accuracy of the classification depends on the classification problem with GMM-EM [10]. Also using the EM algorithm in their research, Leal, et al. [11] gave the observed data set and variable values for the GMM model in the first iteration, with each label c in the initialization cluster k , group of iteration is done by calculating the probability value of each point in each cluster of each iteration j . Then, the parameters for the model are updated and repeated until the convergence [11].

As can be seen, with studies on EM algorithm used as follows:

- An initial prediction is made for model parameters and a probability of distribution is generated. Sometimes called “E-Step” for the “Expected” distribution.
- New data observed based on the model.
- The probability distribution from step E is refined to include new data. This step is sometimes called “M-step”.
- Repeat steps E and M until a stable distribution (convergence) is reached [5].

The main idea of EM is to first create a lower boundary of the likelihood function $l(\theta)$, then gradually increase this margin to increase the value of $l(\theta)$. In Fig. 1, the process of EM algorithm is described. In particular, the black eyebrow curve is a function of $l(\theta)$ and the red eyebrow curve corresponds to the lower bound. There will be many lower bounds of $l(\theta)$. In step E, the algorithm will choose the lower bound close to $l(\theta)$. In step M the lower bound will be maximized. When the value of $l(\theta)$ is greater than the lower bound, $l(\theta)$ will be increased [6].

According to Zhu, et al.[7] it is assumed that similar data from a certain class are made up of Gaussian

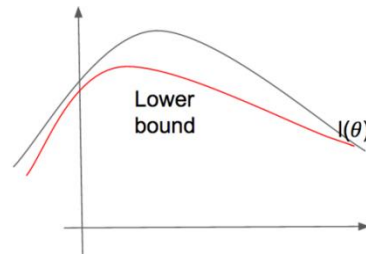


Fig. 1 EM algorithm process.

components. The value of the random variable Y will be determined in the corresponding class. The classification process is applied:

- (1) Estimating GMM model by EM algorithm
- (2) Data classification based on GMM model

The EM classification algorithm does not classify based on distance. The algorithm calculates the probability for each observation belonging to each class based on the chosen distribution, the main purpose of the EM classification algorithm is to find classification solutions to maximize the overall probability for classification data with the required number of classes. Therefore, in the classification problem with EM any difference in the range or scope of the variable selected for the analysis will not affect the classification results [12].

However, the EM algorithm is very sensitive to initialization values and is prone to errors with local minimum. In addition, the algorithm is difficult to focus, and covariance matrices corresponding to one or more components can become an error condition. Wishing to improve one of the EM algorithm’s disadvantages, the author improved the EM algorithm by dividing the point cloud into smaller point clouds vertically, using the pPCA (Probabilistic Principle Component Analysis) model to estimate parameters for GMM (this model is reduced in number of dimensions according to PCA (Probability Component Analysis)), use EM to conduct point cloud classification (applied to each point cloud part) and assess the degree exactly. To improve the algorithm’s convergence time, the scheduling parameter β is used, where β is initialized with a very small value (approximately 0) [13].

2. Material and Methods

2.1 Material

The LiDAR data used in the article were investigated in Uong Bi City of the Quang Ninh Province. This is an area with mountainous terrain that occupies two thirds of the area and hilly slopes inclining from the north to the south. Data were collected with the full waveform airborne LiDAR survey. Data were measured from the 1st of September, 2017 to the 7th of September, 2017, with an area of 297 km², 0.25 pulse/m², and a density of 3.18 points/m². The data include 166.280 points, with $X_{\min} = 420,893.57$, $Y_{\min} = 4,467,227.05$; $X_{\max} = 421,466.87$, $Y_{\max} = 4,467,690.4$. The format of data is “.las”. X, Y, Z values of each point is shown in Fig. 2, the 3D point cloud is shown in Fig. 3.

2.2 Method

Based on the idea of installing EM algorithm, the proposed method is implemented as Fig. 2.

We have LiDAR point cloud $P = \{P_1, P_2, \dots, P_n\}$ with each point P_i in point cloud there is a set of values (X_i, Y_i, Z_i) showing location (X, Y) and elevation (Z) of point.

The value that students use to do classification problems is the value of the height Z_i of the point (classifying points by height). In the point cloud data set with N points, the width of the data $D = 3$. The implementation steps of the method are explained as follows:

(a) *Step 1: Extract the elevation of point*

From the point set in the point cloud, take the point height value as input for the later computation steps,

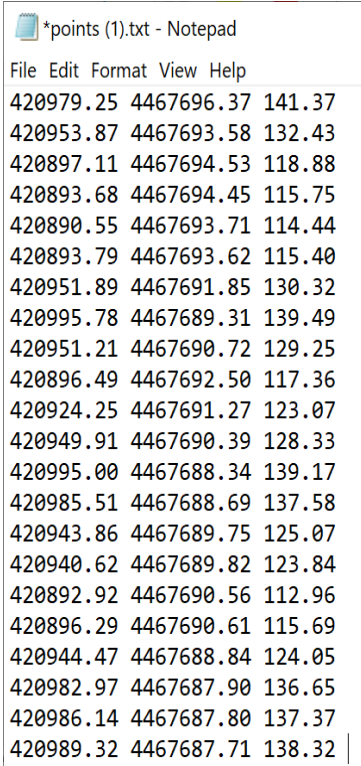
which means we now reduce the width of the data $D = 3 \rightarrow D = 1$.

(b) *Step 2: Initialize model parameter*

Suppose there is a data set $Z = \{Z_1, Z_2, \dots, Z_n\}$ with Z being the point elevation. We have a probability model for the Gaussian distribution shown as follows:

$$P(Z_i) = \sum_{k=1}^K \Pi_k \text{Gaussian}(Z_i | \mu, \Sigma_k, C_k) \quad (1)$$

in which, μ —average value of the data set, Σ —covariance matrix, C —mixing coefficient, σ —standard deviation, β —scheduling parameter (reduces the convergence time of the algorithm EM) [12].



```
*points (1).txt - Notepad
File Edit Format View Help
420979.25 4467696.37 141.37
420953.87 4467693.58 132.43
420897.11 4467694.53 118.88
420893.68 4467694.45 115.75
420890.55 4467693.71 114.44
420893.79 4467693.62 115.40
420951.89 4467691.85 130.32
420995.78 4467689.31 139.49
420951.21 4467690.72 129.25
420896.49 4467692.50 117.36
420924.25 4467691.27 123.07
420949.91 4467690.39 128.33
420995.00 4467688.34 139.17
420985.51 4467688.69 137.58
420943.86 4467689.75 125.07
420940.62 4467689.82 123.84
420892.92 4467690.56 112.96
420896.29 4467690.61 115.69
420944.47 4467688.84 124.05
420982.97 4467687.90 136.65
420986.14 4467687.80 137.37
420989.32 4467687.71 138.32
```

Fig. 2 X, Y coordinate and elevation of points.

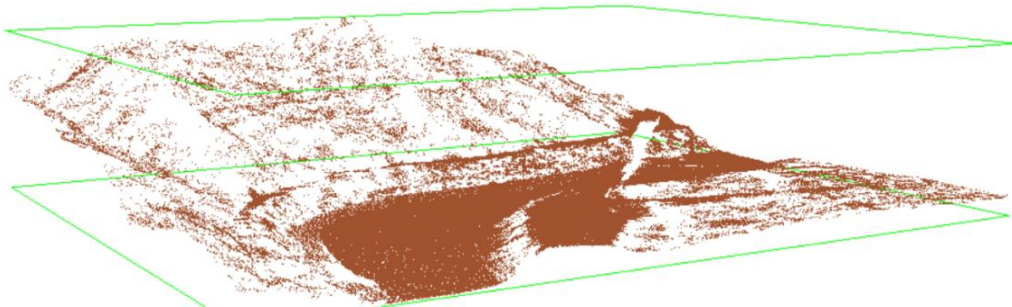


Fig. 3 3D LiDAR point cloud before classification.

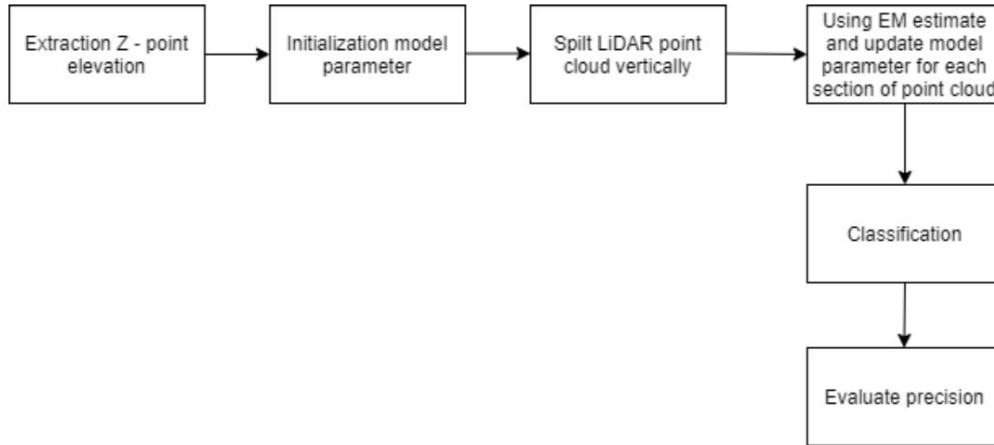


Fig. 4 Proposed method.

(b1) Initialize μ, β

With the value μ after extracting the elevation information of the point in step 1, we will calculate μ by taking the average value of the height of all points in the point cloud according to Eq. (2):

$$\mu = \frac{\sum_{i=1}^n Z_i}{n} \quad (2)$$

β is initialized with a value of approximately 0. The parameter β can roughly be interpreted as the inverse of temperature [13]. At each β value algorithm iterates E step and M step until convergence.

(b2) Initialize Σ and calculate components for model with pPCA

pPCA is used to determine the components of the model and compute the covariance matrix of data. pPCA is a size reduction technique for analyzing data through a lower dimensional latency [14]. It is often used when missing values in data or for multidimensional proportions. PCA is a variation of PCA as a latent variable model. Each data point is assumed to be created as a linear function of the Gaussian latent variables, plus noise [15]. pPCA is often considered to be a latent variable model according to the maximum likelihood method for parameter search through solving the unique value problem of covariance matrix. Covariance matrix is a square matrix of size $m \times m$ (where, m is the number of dimensions of the data). Each element of a matrix is a variance of a data sample [16].

The deviation of each data point from the average value of the symbol r_i is calculated by the formula:

$$r_i = \frac{(Z_i - \mu)}{\sigma} \quad (3)$$

There, σ is standard deviation which is calculated by the formula:

$$= \sqrt{\frac{1}{n} \sum_{i=1}^n (r_i)^2} \quad (4)$$

Covariance matrices are usually computed by formula:

$$\Sigma = \frac{1}{n} \sum_{i=1}^n r_i r_i^T + r_i^2 I_D \quad (5)$$

where, I_D is a matrix with one element = 1, other elements = 0 of D dimensions data.

To calculate the first component number of the model, two values to be calculated are eigenvector and eigenvalue. Eigenvector and eigenvalues are linear algebra concepts that we need to calculate from the covariance matrix to determine the main components of the data. The symbol eigenvalue is λ and e is the eigenvector of the covariance matrix.

Eigenvalue can be calculated by the formula:

$$\det(\Sigma - \lambda I) = 0 \quad (6)$$

in which, I is the orthogonal matrix with the main

diagonal components = 1 and the other components = 0.

Eigenvector is a connection between a linear transformation and covariance matrix. An eigenvector is a vector whose direction does not change when applying a linear transformation to it. It can be expressed as is:

$$\Sigma^* e_i = \lambda_i^* e_i \quad (7)$$

We can transform covariance matrix using eigenvector and eigenvalue according to the formula below:

$$\Sigma_k = V(L - r_i^2 I)^{1/2} R \quad (8)$$

in which, V is the eigenvector matrix of covariance matrix, L is the matrix containing eigenvalue values, R is the orthogonal matrix of the variance values.

The eigenvectors are unit vectors that represent the direction of the largest variance of the data, while the eigenvalues denote the magnitude of this variance in the corresponding directions. This means that R represents a rotation matrix (orthogonal matrix of Σ) and L represents a ratio matrix (diagonal matrix of covariance matrix—the value of the variance of each price, measured value). In order to accurately calculate the number of major components and covariance matrix after conversion, eigenvector is a very important value. And it can be said that the main components of a data set are eigenvectors of the covariance matrix of a dataset.

(b3) Initialize mixing coefficient C_k

The mixing coefficient is the probability to determine the proportion of data belonging to the k component and must ensure conditions:

$$\sum_{k=1}^K C_k = 1 \quad (9)$$

Determining the mixing coefficients for k components of GMM model is calculated by the formula:

$$C_k = \frac{1}{N} \sum_{n=1}^N \gamma_k(Z_i) \quad (10)$$

$\sum_{n=1}^N \gamma_k(Z_i)$ is the total number of points in the dataset and belongs to the k th component. In this study, author initiates a mixing coefficient for all components of the model to be equal.

(c) Step 3: Split LiDAR point cloud vertically

With a very large point dataset in the point cloud, updating parameters and converging calculations takes a lot of time. Therefore, the graduate student divides the point cloud vertically based on the height of the point. However, choosing the number of parts to split to prevent the algorithm from falling into a never-ending state requires calculation.

To solve this problem, the PhD student took an example on a given point data set (number < 3,000), the algorithm converges very well. Therefore, it is recommended that the number of points cloud points to be divided should be less than 10.

(d) Step 4: Use EM estimate model parameter

(d1) Increase value β by 1 unit for each loop (value of β is not greater than 1, $\beta_{\max} = 1$). If $\beta > 1$, reset β value = 1.

(d2) At step E of the algorithm, calculate the probability that a point Z_i belongs to the k th component by the formula:

$$P(k|Z_i) = \frac{(P(Z_i|k)P(k))^\beta}{P(Z_i)^\beta} \quad (11)$$

With,

$$P(Z_i) = \sum_{k=1}^K P(Z_i|k)P(k) \quad (12)$$

and

$$P(Z_i|k) = \sum_{k=1}^K C_k \text{Gaussian}(Z_i|\mu, \Sigma_k) \quad (13)$$

And the Gaussian distribution function is calculated according to the formula:

$$\text{Gaussian}(Z_i|\mu, \Sigma_k) = \frac{1}{2\pi\Sigma_k} * e^{-\frac{r_i^2}{2\sigma^2}} \quad (14)$$

(d3) At step M, update the parameters of the model with the values of μ , Σ , C , σ respectively according to

the following formulas:

$$\Sigma_k = \frac{\sum_{n=1}^N P(k|Z_i)(Z_i - \mu_k)(Z_i - \mu_k)^T}{\sum_{n=1}^N P(k|Z_i)} \quad (15)$$

$$\mu_k = \sum_{i=1}^n \left(\frac{P(k|Z_i) * Z_i}{\sum_{i=1}^n P(k|Z_i)} \right) \quad (16)$$

$$C_k = \frac{\sum_{i=1}^n P(k|Z_i)}{\sum_{k=1}^K (\sum_{i=1}^n P(k|Z_i))} \quad (17)$$

$$\sigma_k = \sqrt{\frac{\sum_{i=1}^n ((Z_i - \mu_k)^2 * P(k|Z_i))}{\sum_{i=1}^n P(k|Z_i)}} \quad (18)$$

(d4) Algorithm convergence: the algorithm converges when the new parameter does not change compared to the parameter that exceeds the threshold value ε , with ε calculated by the following formula:

$$\varepsilon = \sqrt{\frac{1}{n} \sum_{i=1}^n r_i^2} \times 10^{-8} \quad (19)$$

(e) Step 5: LiDAR point cloud classification

Based on the probability of a point belonging to a certain k class in the model, we can proceed to classify points. Based on observed data, if $P(k|Z_i) \geq 0.5$ then conclude that Z_i belongs to class k .

(f) Step 6: Precision evaluating

To evaluate the accuracy of the algorithm, author uses the following measurements: accuracy, recall and F1 measurement of the improved EM algorithm with the basic EM algorithm and the basic EM.

Pseudocode of algorithm can be performed below:

Begin

Input: LiDAR point cloud $P = \{P_1, P_2, \dots, P_n\}$

Output: Point cloud classified

Initialization: the mean μ , covariance matrix Σ , mixing coefficient C , standard deviation σ , scheduling parameter β

Procedure:

for $i = 1$ to n

for $i = 1$ to k

Increase β up to 1 unit

E-step

$$P(k|Z_i) = \frac{(P(Z_i|k)P(k))^\beta}{P(Z_i)^\beta}$$

M-step

$$\Sigma_k = \frac{\sum_{n=1}^N P(k|Z_i)(Z_i - \mu_k)(Z_i - \mu_k)^T}{\sum_{n=1}^N P(k|Z_i)}$$

$$\mu_k = \sum_{i=1}^n \left(\frac{P(k|Z_i) * Z_i}{\sum_{i=1}^n P(k|Z_i)} \right)$$

$$C_k = \frac{\sum_{i=1}^n P(k|Z_i)}{\sum_{k=1}^K (\sum_{i=1}^n P(k|Z_i))}$$

$$\sigma_k = \sqrt{\frac{\sum_{i=1}^n ((Z_i - \mu_k)^2 * P(k|Z_i))}{\sum_{i=1}^n P(k|Z_i)}}$$

if (convergence) stop loop

else repeat E and M step

if $P(k|Z_i) > 0.5$ Z_i belongs to class k

else Z_i does not belongs to class k

Precision evaluating

End

3. Experiment and Discussion

With the data set, the author has tested the classification with the proposed algorithm and the original EM algorithm. The accuracy of the algorithm will be assessed on accuracy, recall and F1 measurement.

Extracting the point height information gives the following result in Fig. 5:

Display points in height and divide the point cloud vertically with 7 parts because the number of points is not too large, but the height distribution of points is uneven, there is a large elevation difference, so the selection of part numbers needs to divide by 7 to avoid uneven height distribution between the parts to help the algorithm on each part converge quickly, then the point cloud is represented as follows in Fig. 6.

The model parameters are initialized on the Z value as follows:

$$\mu = 46.22,$$

$$\Sigma = 276.73,$$

$$\delta = 16.64, \beta = 10^{-8},$$

```
*points (1).txt - Notepad
File Edit Format View Help
141.37
132.43
118.88
115.75
114.44
115.40
130.32
139.49
129.25
117.36
123.07
128.33
139.17
137.58
125.07
123.84
112.96
115.69
124.05
136.65
137.37
138.32
```

Fig. 5 Result of extracting elevation of points.

The eigenvalue value is calculated in Fig. 7:

We have a component matrix of model in Fig. 8:

Thus, with this method, the number of components of the model is determined as 2. The mixing coefficient is initialized equally with 2 components

$$C1 = C2 = 0.5.$$

Using the EM algorithm to recalculate the model's parameters on each part, with 25 iterations, the algorithm converges with the parameters changed in Fig. 9.

Conducting classification with the condition of classification $P(k|Z_i) \geq 0.5$, it can be concluded that Z_i score belongs to class k ($k = 2$). We have a class 1 score of 154,458 points, a class 2 score of 11,822 points. We have the distribution of points into grades 1 and 2 after the classification shown in Fig. 10 and correlation between frequency of distribution and cumulative percentage of data is shown in Table 1.

To evaluate the accuracy of the proposed EM algorithm, the author compared the results with the results of basic EM. The results are shown in Table 2 with precision, recall and F1 measurements. The result is shown in Table 2.

I have tested and compared the proposed algorithm evaluation with the original EM algorithm with improved accuracy and faster runtime by using

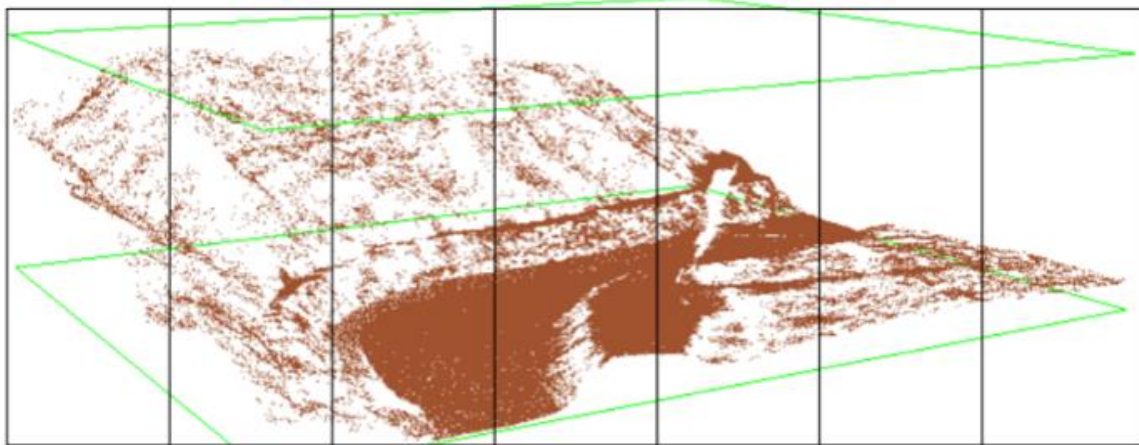


Fig. 6 Split LiDAR point cloud vertically.

		Total Variance Explained		
		Initial Eigenvalues ^a		
	Component	Total	% of Variance	Cumulative %
Raw	1	553.472	100.000	100.000
	2	5.684E-014	1.027E-014	100.000
Rescaled	1	553.472	100.000	100.000
	2	5.684E-014	1.027E-014	100.000

Fig. 7 Eigenvalue of covariance matrix.

Component Matrix^a

	Raw	Rescaled
	Component	Component
	1	1
Z	16.635	1.000
VAR00001	16.635	1.000

Fig. 8 Component matrix of model.

```

Terminal: Local x +
cost: 0.020056501612998545
=====21=====
Gaussian 1:  $\mu = 9.2e+01$ ,  $\sigma = 2.4e+01$ , weight = 0.08
Gaussian 2:  $\mu = 4.2e+01$ ,  $\sigma = 7.5$ , weight = 0.92
cost: 0.011618301854468882
=====22=====
Gaussian 1:  $\mu = 9.2e+01$ ,  $\sigma = 2.4e+01$ , weight = 0.08
Gaussian 2:  $\mu = 4.2e+01$ ,  $\sigma = 7.5$ , weight = 0.92
cost: 0.006731861503794789
=====23=====
Gaussian 1:  $\mu = 9.2e+01$ ,  $\sigma = 2.4e+01$ , weight = 0.08
Gaussian 2:  $\mu = 4.2e+01$ ,  $\sigma = 7.5$ , weight = 0.92
cost: 0.00390137592330575
=====24=====
Gaussian 1:  $\mu = 9.2e+01$ ,  $\sigma = 2.4e+01$ , weight = 0.08
Gaussian 2:  $\mu = 4.2e+01$ ,  $\sigma = 7.5$ , weight = 0.92
cost: 0.002261160989291966
=====25=====
Gaussian 1:  $\mu = 9.2e+01$ ,  $\sigma = 2.4e+01$ , weight = 0.08
Gaussian 2:  $\mu = 4.2e+01$ ,  $\sigma = 7.5$ , weight = 0.92
cost: 0.0013108756393194199
Input Gaussian 1:  $\mu = 8.2e+01$ ,  $\sigma = 4.6e+01$ 
Input Gaussian 2:  $\mu = 8.2e+01$ ,  $\sigma = 4.6e+01$ 
Gaussian 1:  $\mu = 9.2e+01$ ,  $\sigma = 2.4e+01$ , weight = 0.08
Gaussian 2:  $\mu = 4.2e+01$ ,  $\sigma = 7.5$ , weight = 0.92
    
```

Fig. 9 Update parameter for model using EM.

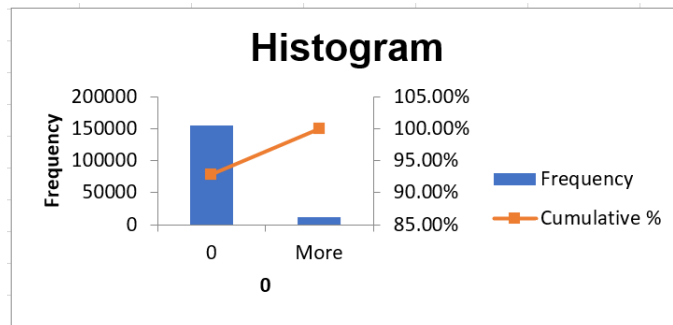


Fig. 10 Histogram of point distribution.

Table 1 Correlation between frequency of distribution and cumulative percentage of data.

1	Frequency	Cumulative (%)	1	Frequency	Cumulative (%)
1	154,458	92.89%	1	154,458	92.89%
2	11,822	100.00%	2	11,822	100.00%

Table 2 Compare result proposed EM with basic EM.

	Precision	Recall	F1	Running time (s)	Convergence
Proposed EM	92.03%	92.06%	0.92045	72.3	0.0013
Basic EM	91.80%	91.79%	0.91795	102.45	0.00021

scheduling parameters to help the algorithm convergent better. However, due to the condition that if $\beta > 1$ consider the condition for $\beta = 1$, the condition for the convergence algorithm is not as good as the original EM algorithm. With the proposed EM algorithm classification results, it meets the requirements of LiDAR point cloud classification, but more research is needed to make the algorithm's convergence conditions better and improve the accuracy.

5. Conclusions

EM algorithm is a popular algorithm in data mining and is used in many different problems. With the LiDAR point classification cloud problem, with the idea of classification based on a one-point posterior probability belonging to the classification class, EM algorithm proposed classification with more than 92% accuracy to meet the requirements of establishment. However, the algorithm needs further development when only classifying the cloud into 2 classes first pulse and last pulse. While the number of unlabeled points is very large, this is a useful amount of information to study the nature and morphology of the object.

With the data set measured in QuangNinh province, Vietnam, the algorithm gives satisfactory results and the running time with an acceptable convergence. However, more data sets need to be tested to verify the accuracy.

Acknowledgment

This research is funded by the Hanoi University of Mining and Geology under project number T19-03 and Ministry of Education and Training number CT.2019.01.07

References

- [1] Zhang, K., Bi, W. H., Zhang, X. M., Fu, X. H., Zhu, K. P., and Zhu, L. 2015. "A New Kmeans Clustering Algorithm for Point Cloud." *International Journal of Hybrid Information Technology* 8 (9): 157-70.
- [2] Chang, Y. C., Fhabib, A., Lee, D. C., and Yom, J. H. 2008. "Automatic Classification of LiDAR Data into Ground and Non-Ground Points." *The International Archives of the Photogrammetry, RS and Spatial Information Sciences* 37 (B4): 457-66.
- [3] Rodriguez-Cuenca, B., Cortes, S. G., Ordonez, C., and Alonso, M. C. 2015. "Automatic Detection and Classification of Pole-Like Objects in Urban Point Cloud Data Using an Anomaly Detection Algorithm." *Remote Sensing* 7: 12680-703.
- [4] Lin, K. F., Wang, C. P., and Sui, P. H. 2012. "Object-Based Classification for LiDAR Point Cloud." [Semanticscholars.https://pdfs.semanticscholar.org/ea05/a9226252f933470a88fa73a3150802ad08e3.pdf](https://pdfs.semanticscholar.org/ea05/a9226252f933470a88fa73a3150802ad08e3.pdf).
- [5] Hui, Z., Cheng, P., Ziggah, Y. Y., and Nie, Y. 2018. "A Threshold-Free Filtering Algorithm for Airborne LiDAR Point Clouds Based on Expectation Maximization." In *Proceedings of the International Archives of the Photogrammetry, RS and Spatial Information Sciences*, pp. 62-3.
- [6] Maligo, A. 2016. *Unsupervised Gaussian Mixture Model for the Classification of Outdoor Environments Using 3D Terrestrial LiDAR Data*. France: Toulouse.
- [7] Zhu, Q., Li, Y., Hu, H., and Wu, B. 2017. "Robust Point Cloud Classification Based on Multi-level Semantic Relationships for Urban Scenes." *ISPRS Journal of Photogrammetry and Remote Sensing* 129: 86-102.
- [8] Lodha, S., Helmbold, D. P., and Fitzpatrick, D. M. 2007. "Aerial LiDAR Data Classification Using Expectation-Maximization." In *Proceedings of the SPIE Conference on Vision Geometry XIV 6499*, 1-11, January 2007.
- [9] Hui, Z. Y., Li, D. J., Jin, S., Ziggah, Y. Y., and Wang, L. 2019. "Automatic DTM Extraction from Airborne LiDAR Based on Expectation-Maximization." *Optics and Laser Technology* 112: 43-55.
- [10] Luo, C., and Sohn, G. 2013. "Line-Based Classification of Terrestrial Laser Scanning Data Using Conditional Random Field." *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL-7/W2, 155-60, <https://doi.org/10.5194/isprsarchives-XL-7-W2-155-2013>.
- [11] Leal, N., Leal, E., and German, S. T. "A Linear Programming Approach for 3D Point Cloud Simplification." *IAENG International Journal of Computer Science* 44 (1): 60-7.
- [12] Ueda, N., and Nakano, R. 1998. "Deterministic Annealing Variant of the EM algorithm." Accessed March 6, 2020. <https://papers.nips.cc/paper/941-deterministic-annealing->

variant-of-the-em-algorithm.pdf.

- [13] Yang, H. L., Peng, J. H., and Zhang, D. X. 2013. "An Improved EM Algorithm for Remote Sensing Classification." *Chinese Science Bulletin* 58 (9): 1060-71.
- [14] Naim, I., and Gildea, D. 2012. "Convergence of EM Algorithm for GMM with Unbalanced Mixing Coefficients." In *Proceedings of the International Conference on Machine Learning*.
- [15] Tipping, M. E., and Bishop, C. M. 1999. "Probabilistic Principal Component Analysis." *Microsoft Research* 61: 611-22.
- [16] E. S. Handbook. "Engineering Statistics Handbook." NIST Sematech, [Online]. Available: <https://www.itl.nist.gov/div898/handbook/pmc/section5/pmc55.htm>. [Accessed 27 2 2020].