Applied Soft Computing Journal 97 (2020) 106716

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

A fast instance selection method for support vector machines in building extraction

Mohammad Aslani [a,*], Stefan Seipel [a,b]

[a] Department of Computer and Geo-spatial Sciences, University of Gävle, Gävle, Sweden
[b] Division of Visual Information and Interaction, Department of Information Technology, Uppsala University, Uppsala, Sweden

ARTICLE INFO

Article history:
Received 10 March 2020
Received in revised form 24 July 2020
Accepted 8 September 2020
Available online 11 September 2020

Keywords:
Support vector machines
Data reduction
Instance selection
Big data
Building extraction

ABSTRACT

Training support vector machines (SVMs) for pixel-based feature extraction purposes from aerial images requires selecting representative pixels (instances) as a training dataset. In this research, locality-sensitive hashing (LSH) is adopted for developing a new instance selection method which is referred to as *DR.LSH*. The intuition of *DR.LSH* rests on rapidly finding similar and redundant training samples and excluding them from the original dataset. The simple idea of this method alongside its linear computational complexity make it expeditious in coping with massive training data (millions of pixels). *DR.LSH* is benchmarked against two recently proposed methods on a dataset for building extraction with 23,750,000 samples obtained from the fusion of aerial images and point clouds. The results reveal that *DR.LSH* outperforms them in terms of both preservation rate and maintaining the generalization ability (classification loss). The source code of *DR.LSH* can be found in https://github.com/mohaslani/DR.LSH.

1. Introduction

Support vector machines (SVMs) [1] are among the most powerful supervised classifiers in machine learning that have been abundantly applied to a multitude of pattern recognition problems in remote sensing and geoscience [2–4]. Structural risk minimization (minimizing the misclassification risk on unseen data) and convex quadratic optimization are two features that distinguish SVMs from other classifiers in terms of classification ability [5].

The quality of the remote-sensing-derived products constructed by SVMs (or generally all classifiers) relies on the classification accuracy [6] which mainly depends on the samples selected to be used in the training phase [7,8]. The number and coverage of training data (training pixels; a pixel as a sampling unit) in the feature space should be such that they suitably represent the entirety of classes [9]. The value of training data is a function of many factors of which the spatial variability of spectral signatures of classes is a key factor [7]. In fact, the training data should effectively cover all the crucial regions in the feature space so that accurate classification is assured. Effectively covering the crucial regions in the feature space usually leads to redundancy and training data expansion especially in high-resolution images due to the high correlation of neighboring pixels in the feature space. Also, manually identifying a small subset of only the effective pixels in the image space is a nontrivial and time-consuming task.

Redundancy and increase in the number of training samples significantly reduce the speed of SVM in the training phase due to the high time and memory complexity of its training [10]. In other words, the high computational and memory complexities of SVM make it impractical and can hinder achieving the results while handling a huge number of pixels. With the aim of tackling this issue, researchers have proposed several methods to expedite the learning process of SVM facing with huge datasets. These methods are classified into two categories. In the first category, methods aim at accelerating the calculation of SVM training by either reducing the complexity of the underlying optimization problem (e.g., decomposition of the original quadratic problem to the problems with smaller size) or speeding up the optimization problem [11–14]. The second category includes data reduction methods to reduce the computational complexity in classification [15,16]. These methods try to preserve the generalization ability of SVM by choosing a specific number of effective samples and features [8]. They assist in inordinate memory and time complexities in dealing with the contemporary huge datasets. Theoretically, the methods in the first category do not have the required efficiency for handling a huge number of training data and they still need huge memory space [17,18]. Whereas, there are a few advantages associated with the second category.

* Corresponding author.
E-mail addresses: mohammad.aslani@hig.se (M. Aslani), Stefan.Seipel@hig.se (S. Seipel).

https://doi.org/10.1016/j.asoc.2020.106716

First, data reduction methods are usually more general than the methods in the first category and may be beneficial in different classifiers. Second, the data become less demanding in terms of the required memory.

Among manifold approaches in data reduction (second category), instance selection has drawn the attention of researchers owing to the growing number of records in datasets [19]. Instance selection methods strategically select a manageable subset of training data that are representative of the original set [20,21]. They reduce the data size and speed up the training process by discarding the instances that do not significantly contribute in classification accuracy [22,23]. The focus of this research is on extracting the effective training data such that the original pattern is preserved (i.e., instance selection).

Within such a context, a myriad of instance selection methods have been proposed. The methods that yield more efficient results usually suffer from fairly high computational complexity. This makes these methods slow or almost impractical while handling huge datasets with several millions of records. Therefore, the need for the methods that are both fast (linear time complexity) and efficient (low value of loss or high value of accuracy) is indispensable in huge training datasets.

In this research, a new fast instance selection algorithm inspired by locality-sensitive hashing (LSH), owing to its suitable computational complexity, is presented. In this algorithm, the instances whose degree of similarity to a given sample is greater than a pre-established similarity threshold are quickly identified and removed. The remarkable features of the proposed algorithm are:

- Simplicity
- Linear time complexity ($O(n)$)
- Integer-based calculations
- Satisfactorily preserving the extent of known classes
- Easy control of the preservation rate by adjusting the input parameters especially similarity threshold.

The remaining part of this paper is organized as follows: Section 2 reviews instance selection methods and summarizes the gaps in the existing literature. Section 3 introduces the principles of LSH. Section 4 describes the operation of the proposed method and its behavior in detail. In Section 5, a dataset for building extraction from the fusion of high-resolution aerial images and point clouds is prepared. The results are presented and discussed in Section 6. Finally, Section 7 concludes the paper and proposes some directions for future work.

2. Related work

Manifold instance selection methods have been proposed in the context of classification and a comprehensive survey of the existing methods in this realm may be found in [19,23,24]. These methods can be classified by using a range of different taxonomies and there is no comprehensive and universally accepted categorization of all instance selection methods. Nevertheless, most of them, based on the underlying method, can be categorized into random-based, clustering-based, distance-based, neighborhood-based, and tree-based methods. It should be noted that evolutionary approaches are not reviewed in this paper due to their high computational complexity, although they deliver outstanding results in terms of accuracy and reduction rate [25].

The random-based methods select a small subset of instances in a random manner to form the reduced training dataset. Simplicity, data size independence, and very low computational complexity are the biggest advantages of such methods that make them feasible for many different types of datasets [21,26,27]. However, those methods might not be able to adequately represent all classes. Also, they might deliver a high standard deviation of classification accuracy and loss [28].

In the clustering-based methods, after a given clustering technique is applied, the clustered samples that lie far away from decision boundaries are removed, and the refined samples from the crucial clusters are selected [29]. In other words, there are generally three steps in clustering-based methods: 1- clustering the data, 2- identifying the crucial clusters, and 3- detecting influential instances. Frequency sensitive competitive learning [29], k-means [30–32], fuzzy clustering [33], adaptive clustering [34], minimum enclosing ball [35], hierarchical micro-clustering [36], and Ward-linkage clustering [37] are different clustering techniques that have been used in previous research.

The clustering step can be done for instances of each class separately [36–38] or for all instances regardless of their classes [30–32,35,39]. Regarding selecting the crucial clusters, the authors in [30] and [39] considered homogeneous clusters (those that contain instances of a single class) as ineffective ones that do not contribute much to extracting the border of classes and preserved only one point from each of them (centroid or a point near the centroid). By way of contrast, the methods proposed in [31,32,35] do not ignore all the homogeneous clusters as they might contain boundary instances. It was shown that replacing each homogeneous cluster with only one point may impair the decision functions. In [32,35], all the heterogeneous clusters and the homogeneous clusters that are near the decision boundaries are selected for further instance selection analysis. Different methods have been applied for selecting potentially valuable samples from crucial clusters such as using a rough sketch of decision boundaries obtained by an initial SVM [36], safety region [31,38], convex–concave hull [40], and Fisher's discriminate analysis [32]. One of the difficulties of the clustering-based methods is a proper selection of the influential parameters such as the number of clusters. Also, the clustering performance has a direct bearing on the selected samples [41].

In the distance-based methods, the points near the opposite class [27,42,43] or densest points [44] are detected by calculating Euclidean, Mahalanobis, or Hausdorff. Calculating distances between instances makes these methods computationally high demanding for huge datasets. Moreover, it might not be possible to entirely load all the calculated distances in memory since distances are decimal numbers and occupy much memory. In the method proposed in [44], computing the density of a sample requires calculating Euclidean distances of the sample to others, making the method, consequently, slow when applied to huge datasets. In calculating the Mahalanobis distance [42], the complexity of estimating the inverse covariance matrix significantly grows as the dimension of the dataset rises.

In the neighborhood-based methods, effective points are extracted by calculating the statistical properties of their neighbors. The method proposed in [45] hinges on the fact that the samples situated near the decision boundaries have neighbors that are more heterogeneous. The authors suggested two indexes: proximity and correctness. Proximity measures the diversity of the neighbors' class (i.e., heterogeneity of the neighbors) and correctness, as an index for detecting noisy samples, measures the consistency of an instance with its neighbors (i.e., a noisy sample tends to belong to a different class from its neighbors). Calculating k-nearest neighbors, proximity alongside correctness for all the samples increases the computational complexity of this method. The method was improved in [41] such that k-nearest neighbors, proximity, and correctness are calculated only for a subset of the training dataset that is mostly positioned near the decision boundaries. The search space reduction is based on the idea that the neighbors of a sample that is located near the decision boundaries are also placed near the decision boundaries. The

performance of this method is highly a function of the number of neighbors (k). Wang and Kwong [46] integrated the proximity index with active learning for selecting informative samples. In [47], Bayes posterior probability is employed to recognize the points that are close to the decision surfaces. If the estimated class of the sample, calculated by Bayes posterior probability, is not the same as the observed class (true class) of the sample, then it is removed from the training set. The authors indicated that their method is effective for different types of classifiers. Zhu et al. [48] used k-nearest neighbors to develop a new instance selection framework named NearCount. In NearCount, the importance of an instance is determined based on the number of times that it is selected as the nearest neighbor of instances from opposite classes. Naturally, instances that have a zero cited count (not selected as the nearest neighbors) are considered as inner ones and removed from the dataset.

In tree-based methods, extraneous instances are filtered based on the output of decision trees. Guo et al. [49] used bagging [50] as a variant of ensemble learning that combines multiple weak learners. The Classification and Regression Trees (CART) was used as a base-classifier because of its simplicity [51]. The authors proposed an index called ensemble margin that integrates the first and second most voted classes [52]. The samples with small values of ensemble margin are close to the decision surfaces and should be preserved. The results indicated that the proposed method outperforms a random-based method. However, the proposed method is not efficient in handling a huge dataset with a large number of features. Thus, Guo and Boukir [17] proposed to use a very small sampling ratio and ensemble size in bagging to make the algorithm faster without affecting its performance because the aim of using ensemble learning is instance selection rather than classification. Chang et al. [10] used a binary C4.5 algorithm [53] to decompose the original data space into a number of smaller heterogeneous and homogeneous regions. The labels of the samples that fall inside the heterogeneous areas are determined by SVMs. In this way, the computational cost of training SVMs becomes less expensive as SVMs are required to be trained only on small datasets. Another tree-based strategy of approximating decision boundaries proposed in [22] trains SVM only on the samples located near the approximate decision boundaries. The initial SVM trained on a small subset of the original dataset and the decision tree trained by C4.5 algorithm estimate decision boundaries.

Due to the fairly high computational cost of the most above-mentioned methods that makes them infeasible to deal with huge datasets with several millions of records, two neighborhood-based instance methods with linear time complexity were proposed recently: Prototype Selection based on Dense Spatial Partitions (PSDSP) [54] and Instance Selection based on Locality-Sensitive Hashing (LSH-IS-S) [55]. PSDSP aims at selecting paramount samples from the densest partitions obtained by an n-dimensional regular grid. The algorithm has three steps: 1- partitioning the space into non-overlapping regions by using a regular grid, 2- calculating the density of each partition according to its corresponding number of samples, and 3- extracting the indispensable samples from the first k densest partitions. The input parameter k shows the desirable number of samples. The authors indicated that their algorithm achieves reasonable accuracy in 15 different datasets. LSH-IS-S removes redundant (very similar) instances by using LSH [56]. LSH was invented for the aim of enhancing the nearest neighbor search [57]. LSH technique hashes instances such that similar instances fall into the same bucket with high probability (please refer to Section 3 for more information about LSH). LSH-IS-S irregularly divides the feature space into a number of buckets in different layers by using a set of hash functions families. If the bucket assigned to an instance was not occupied by another instance before, the instance is preserved. In this way, only one instance in a group of very similar instances is preserved. The authors benchmarked LSH-IS-S against seven instance selection methods and concluded that it leads to acceptable results in terms of classification accuracy and reduction rate.

On the whole, most of the existing methods either have high time complexity which makes them infeasible in handling huge datasets (e.g., distance-based methods) or have low performance (e.g., random-based methods). In this paper, a new method that has suitable time complexity and performance in facing with a huge dataset is developed. The method inspired by LSH is developed from a perspective based on a similarity index. Since LSH is employed for designing a method for data reduction, the proposed method is referred to as *DR.LSH*. It quickly identifies and removes similar samples to a given sample according to a defined similarity threshold. Therefore, by reducing the number of similar training samples, the training phase of SVM becomes faster without significantly compromising the generalization ability. Unlike clustering and distance-based methods that are computation-intensive particularly in huge training datasets due to calculating the distance between each pair of samples, the proposed method is fast and has low memory consumption because of its linear time complexity and integer-based computations. Taking advantage of eliminating redundant samples (similar samples), the extent (both edges near the opposite class and near the free space) of known classes in the feature space is preserved and thus, removing the fusion of samples from unseen classes will be feasible. In contrast to PSDSP and LSH-IS-S, the similarity of samples is directly measured and considered in the instance selection process of *DR.LSH*.

The performance of *DR.LSH* is benchmarked against LSH-IS-S and PSDSP as two newly developed methods with linear time complexity. Their time complexity enables them to handle huge datasets and provide a fair comparison. The assessment is carried out by using Pareto optimality [58] by considering a reduction rate and a classification loss simultaneously. Also, the knee point concept [59] is utilized for choosing the best configurations of the input parameters of *DR.LSH*.

3. Locality-sensitive hashing: LSH

LSH is a fast method for finding approximate k-nearest neighbors by hashing data into buckets to avoid comparing all pairs of points [56]. The buckets are obtained by some locality preserving hash functions. The idea behind LSH is that the probability for close samples to be projected into the same bucket is higher than for samples that are further. Within such a context, a family $H = \{h : R^n \rightarrow N\}$ that is locality-sensitive (i.e., closer samples have a higher probability to collide, while more further samples have a lower probability to lie in the same bucket) should be used for hashing the data into buckets. In this paper, the following locality preserving hash function is used for hashing each sample ($\overrightarrow{x}$) [60]:

$$h_{\overrightarrow{a},b}(\overrightarrow{x}) = \left\lfloor \frac{\overrightarrow{a} \cdot \overrightarrow{x} + b}{r} \right\rfloor \quad (1)$$

In Eq. (1), $\overrightarrow{a}$ is a d-dimensional vector whose elements are independently chosen from a Gaussian distribution with mean 0 and standard deviation 1. The parameter b is a random real value drawn uniformly from $[0, r]$ and the parameter r controls the width and number of buckets. A small value of r leads to a large number of buckets with a small interval size. Given a specific value of r, the hash function projects sample $\overrightarrow{x}$ along the random direction identified by $\overrightarrow{a}$, then adds a random shift of b, and finally rounds down the obtained value divided by r. For the sake

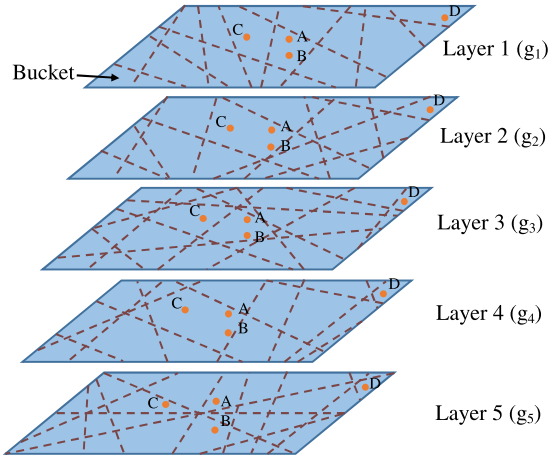

Fig. 1. An example of a similarity index. Different hyper-planes in each layer are the representations of h_i.

of simplicity, all the samples ($\vec{x}$) are normalized into [0, 1] and the parameter r is set to 1.

A set of k independently chosen hash functions of H forms a family of hash functions $g(\vec{x}) = (h_1(\vec{x}), h_2(\vec{x}), \ldots, h_k(\vec{x}))$. In order to achieve high accuracy of search and to increase the probability of success (i.e., the collision probability of close samples in buckets), a set of hash functions families $G = \{g_1, g_2, \ldots, g_l\}$, rather than only one hash functions family, needs to be constructed. In the next section, the concepts of LSH are used for defining the nature of the proposed instance selection method (*DR.LSH*).

4. The proposed instance selection method

This section aims at explaining the way that *DR.LSH* works. The pseudocode of *DR.LSH* along with the detailed explanations of its steps are first presented. Then, the effects of the input parameters on its performance are illustrated.

4.1. DR.LSH

DR.LSH is based on the perspective that there are some redundant samples that do not significantly contribute to the description of each class. Its main goal is to quickly find and remove them in huge datasets. *DR.LSH* accelerates the calculations of finding redundant samples by using the local sensitivity concept. It identifies redundant samples by measuring similarity based on partitioning the space into many buckets in several layers. The redundant samples, relative to a given sample (A), are defined as the samples whose similarity to sample A exceeds a pre-established threshold. *DR.LSH* is composed of two main steps: 1- hashing data points into buckets, and 2- measuring the similarity and removing similar samples (redundant samples).

In the first step, a set of hash functions families is used to identify the buckets that each instance belongs. Let $G = \{g_1, g_2, \ldots, g_l\}$ be a set of families of hash functions such that $g(\vec{x} \in T) = (h_1(\vec{x}), h_2(\vec{x}), \ldots, h_k(\vec{x}))$, and h_i is a hash function $\{h : R^d \rightarrow N\}$. $T = \{(\vec{x_1}, C_1), \ldots, (\vec{x_n}, C_n)\}$ is a training dataset where ($\vec{x}$, C) is an input–output pair, $\vec{x}$ is an d-dimensional input feature vector, C is the output class, and n is the number of training data. Each family of hash functions (g_i) splits the input space into a set of smaller regions called buckets. The output of each hash functions family can be depicted as a layer of non-overlapping d-dimensional buckets. Also, the output of a set of hash functions families (G) with l members can be regarded as l layers of buckets (Fig. 1). Indeed, samples are projected from an d-dimensional decimal space to an l-dimensional integer space such that each feature in the new space is the bucket id in its corresponding layer. This property enables *DR.LSH* to scale well with data dimensionality. It should be noted that a bucket id can be thought of as a unique number assigned to each bucket in each layer.

In the second step (Fig. 2), first, the samples that share a bucket (i.e., they are approximately close), to a given sample, are retrieved and their similarity is measured. A similarity index between two samples (A and B) is defined as the number of identical buckets of A and B in all l layers. In other words, the similarity between two samples is measured based on the number of matching feature values in the new space. Closer samples are more likely to be hashed in the same buckets and thus, they have a high similarity index. Finally, the samples that have a high value of similarity index to a given sample (i.e., bigger than a defined similarity threshold) are considered as superfluous ones and are removed. *DR.LSH* is fast because it does not need to compute a large number of dissimilar pairs.

Fig. 1 shows an example of five layers of hash functions families (g). Let $T = \{A, B, C, D\}$ be the dataset with only one class and A is the current sample. Regarding the similarity index, B and C are the only samples that share a few buckets with A and accordingly, they are considered as the samples that are likely extraneous. The number of identical buckets of A and B is 4 and that of for A and C is 2. Thus, the similarity index between A and B is 4 (out of 5) and the similarity index between A and C is 2 (out of 5).

The pseudocode of *DR.LSH* is presented in Algorithm 1. Data reduction is carried out for each class separately and thus, it is readily applicable for datasets with multiple classes. The inputs of the algorithm are a set of hash functions families (G), a training dataset (T), and a similarity threshold (ST). The algorithm consists of two major loops. In the first loop (lines 2–7), the bucket id of each instance in all layers is determined and recorded independently. The time complexity of the first loop is linear with regard to the number of instances (n) because every instance is included only once.

In the second loop (lines 8–26), first, the samples of class C that have at least one common bucket with the current sample ($\vec{x}$) in all layers are retrieved and stored. Then, the samples whose number of shared buckets with $\vec{x}$ is more than ST are regarded as extraneous instances and removed. Since *DR.LSH* starts with the original dataset and gradually discards those samples that are unnecessary (lines 20–21), it is considered as a decremental method in terms of the order in which instances are processed [24] and thus, its execution time mainly depends on the number of preserved samples. That is, the more samples are removed, the fewer samples are required to be passed through, and the shorter the execution time will be. The time complexity of the second loop is linear with respect to the number of instances (n). Please refer to Appendix for the proof of linearity of *DR.LSH*'s time complexity.

The number of layers (l), number of hash functions in each layer (k), and similarity threshold (ST) are the parameters that directly contribute to *DR.LSH*'s performance (e.g. reduction rate) and its computational cost. The higher value of these parameters (k and l) leads to a longer computation as more fine buckets are created. In addition, the bucket resolution along with the accepted degree of similarity are controlled by k. The parameter k controls the tradeoff between larger bucket size with a lower degree of similarity and smaller bucket size with a higher degree of similarity. The parameter ST significantly affects the reduction rate — in other words, the number of preserved instances increases with the value of ST.

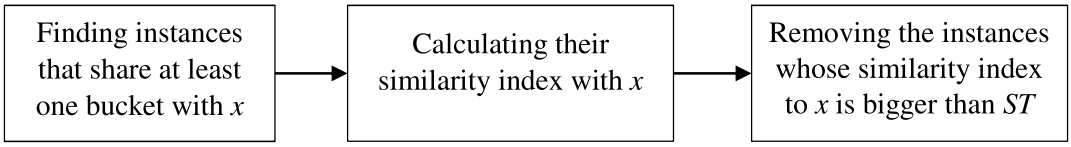

Fig. 2. The workflow of the second step of *DR.LSH* for an instance x.

Algorithm 1 *DR.LSH*, the proposed instance selection algorithm

Input: A training dataset $T = \{(\overrightarrow{x_1}, C_1), \ldots, (\overrightarrow{x_n}, C_n)\}$
A set of hash function families $G = \{g_1, g_2, \ldots, g_l\}$ such that
$g(\overrightarrow{x} \in T) = (h_1(\overrightarrow{x}), h_2(\overrightarrow{x}), \ldots, h_k(\overrightarrow{x}))$ and h_i is a hash function $(h : R^d \rightarrow N)$
ST (similarity threshold) that should not be greater than l
Output: The set of selected instances $(SI \subseteq T)$

$SI \leftarrow \varnothing$
for each $\overrightarrow{x} \in T$ **do**
  for each function family $g \in G$ **do**
    $bid \leftarrow$ bucket id assigned to $\overrightarrow{x}$ by g
    Add bid to b_x^g
  end for
end for
for each class $C \in \{C_1, C_2, \ldots, C_m\}$ **do**
  $S \leftarrow \{\overrightarrow{x_i} | C_i = C\}$; S contains all the instances in class C
  for each $\overrightarrow{x} \in S$ **do**
    $I \leftarrow \varnothing$
    for each function family $g \in G$ **do**
      $bid \leftarrow b_x^g$
      $Neighbors \leftarrow$ all instances of class C in bid except $\overrightarrow{x}$
      Add $Neighbors$ to I
    end for
    for each unique $z \in I$ **do**
      $SimilarityIndex \leftarrow$ calculate the frequency of z in I
      if $SimilarityIndex >= ST$ **then**
        Remove z from S
        Remove z from all buckets
      end if
    end for
  end for
  Add S to SI
end for

4.2. Behavior of DR.LSH

Discarding redundant samples leads to curtailing training data, speeding up the training process, and satisfactorily preserving data extent of each known class (both edges near the opposite class and the free space). The degree of preserving the original extent of classes is controlled by the reduction rate that is itself affected by l, k, and ST. To understand the effect of input parameters on the behavior of *DR.LSH*, we illustrate it by an empirical comparison. In this comparison, a synthetically generated dataset formed by 9000 instances with two numerical features and two classes is used. In this dataset, the number of instances in each class is roughly the same. Fig. 3(a) shows the distribution of the original dataset. It is vivid that the instances are not uniformly distributed and there are some regions in both classes with a higher density of points. Also, there is an overlap between classes.

Table 1 summarizes the number of samples selected by *DR.LSH* when the number of hash functions in each layer (k) and similarity threshold (ST) gradually decline. The number of layers (l) is set to 10 for simplicity. The reduction rate increases as the buckets become coarser and similarity threshold becomes smaller. In fact, the high reduction rate is attributable to the high value of parameters k and ST. This is a useful and interesting feature for the users as they can readily control the reduction rate and degree of preserving the data. Fig. 3 indicates the preserved instances when $k = 20$, $l = 10$, and $ST = 2, 4, 6$, and 8. It is evident that the extent of each class is convincingly maintained depending on the value of ST. This feature helps to feasibly recognize data from unseen classes. Another property of *DR.LSH* is that it makes the data more homogeneous in terms of density. Table 2 compares the execution time (sec) of *DR.LSH*, programmed in MATLAB 2019[1] and run on a computer with an Intel(R) Core™ i7-7700 CPU @ 3.6 GHz and 32 GB RAM, for different values of k and ST when $l = 10$. The longest execution time is 0.176 (s) that is insignificant in handling 9000 points. It is seen that the computational time of *DR.LSH* diminishes as k and ST declines. This is because the number of buckets, the required degree of similarity, and consequently the number of instances selected are reduced.

[1] The source code of *DR.LSH* can be found in https://github.com/mohaslani/DR.LSH.

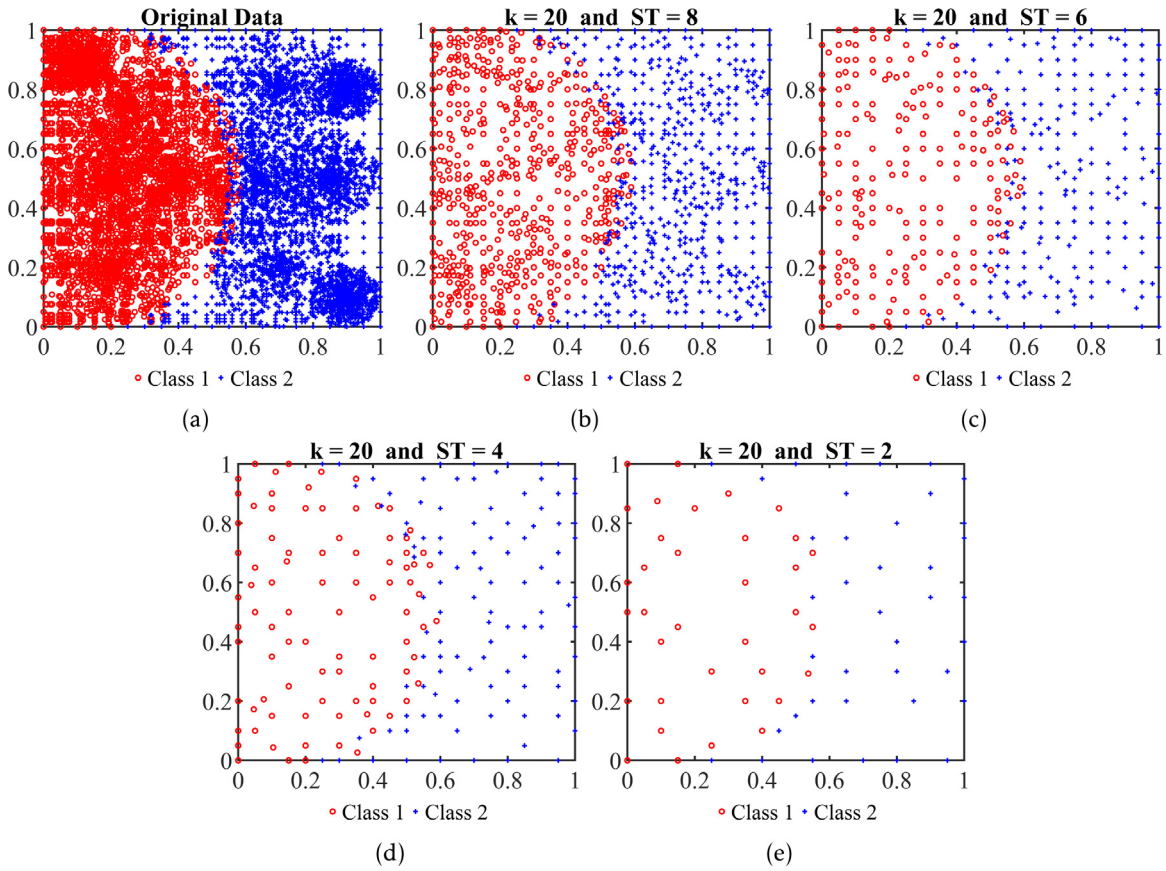

Fig. 3. Selected instances by *DR.LSH*.

Table 1
The number of instances that *DR.LSH* selects for different values of k and ST ($l = 10$)

Similarity threshold (ST)	Number of hash functions in each layer (k)					
	30	25	20	15	10	5
8	2084	1661	1216	810	432	146
6	851	646	462	289	151	52
4	368	277	191	120	64	25
2	146	111	79	51	29	12

Table 2
The execution time (sec) of *DR.LSH* for different values of k and ST ($l = 10$)

Similarity threshold (ST)	Number of hash functions in each layer (k)					
	30	25	20	15	10	5
8	0.176	0.140	0.112	0.086	0.064	0.043
6	0.106	0.091	0.080	0.068	0.052	0.036
4	0.092	0.081	0.071	0.060	0.044	0.032
2	0.082	0.073	0.062	0.052	0.040	0.029

5. Building extraction from the fusion of high-resolution aerial images and point clouds

To verify the effectiveness of the proposed method, it is incorporated in the procedure of automatic building extraction. Automated building extraction from high-resolution aerial images and point cloud has been an important realm of research in remote sensing over the past two decades. Different approaches that have their own advantages and disadvantages have been proposed for building extraction [61,62]. In this context, SVM as a supervised learning algorithm has been widely and successfully utilized in pixel-based classification methods for building extraction because of its good performance [63–65].

The SVM-based building extraction method employed consists of the following five steps: 1- Data preprocessing that aims at removing noise from point clouds (Section 5.2), 2- Feature production whereby features such as height from the ground and vegetation index are derived (Section 5.2), 3- Instance selection (Section 6), 4- Training SVM and optimizing the hyperparameters (Section 6), and 5- Post-processing for supplementing incomplete buildings and eliminating non-buildings (Section 6).

All the five steps affect the quality of the output. However, the worthiness of the maps constructed by SVM is mainly a function of classification performance among others. The performance of the SVM depends on the samples used for its training. Although including all the pixels of the training parts leads to a complete cover of spectral variability and extent of each known class by

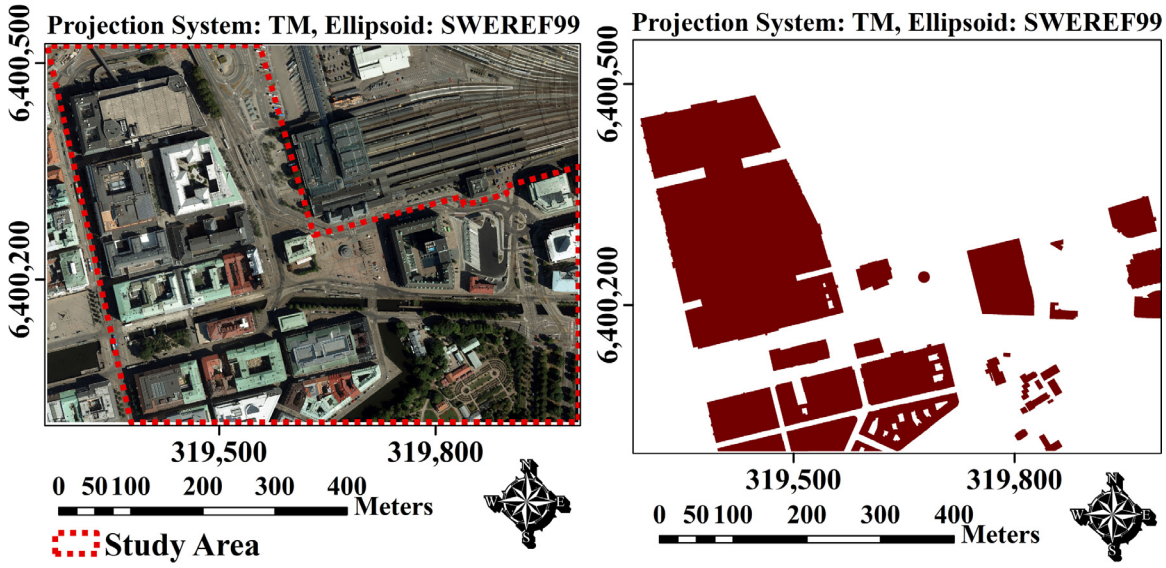

Fig. 4. (a) True-Orthophoto and boundary of the study area and (b) Footprint of buildings.

SVM, this can bring about a huge number of redundant training data. This is especially true in high-resolution images, because of the analogous feature values and/or spatial correlation of neighboring pixels. This can hinder obtaining the results by SVM due to its high time complexity. Therefore, selecting a small subset of training pixels is indispensable to circumvent this issue. In this paper, *DR.LSH* is used to select the pixels that provide a representative description of the overall population in the feature space.

5.1. Study area and data description

The study area is a small part of Gothenburg that is the second-largest city in Sweden. The data for this research provided by the Swedish mapping, cadastral and land registration authority (Lantmäteriet)[2] include True-Orthophoto, image-derived DSM cloud produced by image matching of aerial images, LiDAR data, and the footprint of buildings. The aerial images were acquired in 2018 with the spatial resolution of 10 cm in 4 bands of near-infrared (*NIR*), red (*R*), green (*G*), and blue (*B*) by a digital camera UltraCam Eagle Mark 1[3] with the focal length of 79.8 mm and the pixel size of 5.2 μm. The spatial resolution of both the DSM cloud and True-Orthophoto is 10 cm. The point density of LiDAR data captured by Leica Geosystems ALS60 is 1 point per square meter. All the collected data were georeferenced in SWEREF99. Fig. 4 shows True-Orthophoto and the buildings' footprint of the study area.

5.2. Data pre-processing and features production

In the preprocessing stage, noise as an unavoidable phenomenon that influences outputs is removed from LiDAR data. LiDAR points whose last return (LR) is significantly bigger than their first return (FR) are considered as noises and are eliminated. Image-derived DSM cloud is enhanced by the first echo of each pulse of LiDAR data to make sure that there is no gap in the DSM. In order to train an SVM and enable it to detect buildings, different features of each pixel as the inputs of the classification process are required to be produced. Gradient magnitude, second spatial derivative (Laplacian), terrain roughness, and nDSM are four features that are derived from DSM and normalized difference vegetation index (NDVI) is a feature produced from True-Orthophoto.

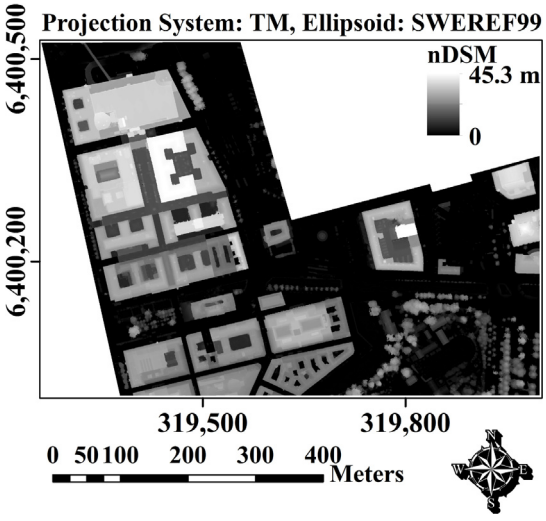

Fig. 5. nDSM.

The digital terrain model (DTM) is generated by using the progressive morphological filter developed in [66]. This method removes non-ground features by gradually increasing the filter size and applying opening operations in each step. The nDSM, which includes the heights of non-terrain objects (e.g., buildings and vegetation), is obtained by subtracting DTM from DSM (Fig. 5).

The gradient magnitude that measures the local change of the height values in the DSM is calculated by Eq. (2). In this equation, $\frac{\partial F}{\partial x}$ and $\frac{\partial F}{\partial y}$ are gradients in x and y directions respectively. Since it is not necessary to find the exact locations of the edges and because of the simplicity, the Sobel operator with odd dimensions (3×3) is used to calculate the gradients in x and y directions [67].

[2] https://www.lantmateriet.se/
[3] http://www.vexcel-imaging.com/

$$Gradient\ Magnitude = \sqrt{(\frac{\partial F}{\partial x})^2 + (\frac{\partial F}{\partial y})^2} \quad (2)$$

The second derivative that enhances the discontinuity of the height values of the DSM is estimated by a 3 × 3 Laplacian kernel Eq. (3).

$$\nabla^2 F \approx F(X+1, Y)+F(X-1, Y)+F(X, Y+1)+F(X, Y-1)-4F(X, Y) \quad (3)$$

In Eq. (3), X and Y are the centers of the Laplacian kernel. Terrain roughness that describes the irregularity of the DSM is calculated by the method proposed in [68]. This method is based on measuring the amount of elevation difference among neighboring cells of a DSM. NDVI as a widely used vegetation index is derived from True-Orthophoto by Eq. (4) where *NIR* and *R* are the reflectances of the near-infrared and red bands respectively.

$$NDVI = \frac{NIR - R}{NIR + R} \quad (4)$$

The above-mentioned features along with the pixels' labels extracted from the reference data (footprint of buildings) are used to form the elements of training data. The size of the training dataset is approximately 23,750,000 records with five input features and one output (building/non-building). Each output class occupies a specific region in the feature space, depending on the intrinsic property and environmental context of the classes.

6. Results and discussion

In this section, the effects of the parameters of *DR.LSH* on its overall performance by using the dataset obtained for building extraction are assessed. Moreover, its performance is compared with two recently proposed instance selection methods, namely LSH-IS-S and PSDSP. These methods are selected for benchmarking because of their linear time complexities that make them applicable to handle several millions of instances.

The performance of an instance selection method can be assessed qualitatively by visual inspection or quantitatively with some metrics. In this research, two quantitative metrics (Eqs. (5) and (6)) are used to assess and benchmark the performance of the aforementioned instance selection methods. Fig. 6 indicates the workflow used for evaluating the instance selection methods. First, the dataset is normalized between 0 and 1. Then, the normalized dataset is partitioned into *outer training* and *outer test sets* in a stratified p-fold cross-validation scheme. Next, an instance selection method (e.g., *DR.LSH*) is applied to the *outer training set*, producing the *selected data*. At this point, the *preservation rate* is measured by the number of the *selected data* and the *outer training set*. Afterward, the *selected data* is used for hyper-parameter optimization (through a stratified cross-validation scheme) alongside training an SVM with the optimized hyper-parameters. Finally, the classification loss is measured on the *outer test set*.

As is evident from Fig. 6, a repeated stratified p-fold cross-validation scheme [69] is employed to estimate the average classification loss and preservation rate. Utilizing test sets for estimating the classification loss through a repeated stratified p-fold cross-validation scheme allows for an unbiased evaluation of the generalization capabilities of an SVM trained on the reduced data. In addition, this scheme results in a lower variance in comparison to a non-repeated version [70]. In this scheme, the parameter p as the number of folds and the parameter r as the number of repeats are set to 10 and 7 respectively. That is, the whole process is repeated 7 times for 10 folds and the final classification loss and preservation rate are obtained by averaging folds and repetitions together.

Since classification loss, as an evaluation metric, significantly relies on the value of hyper-parameters, namely kernel type (RBF, polynomial, and linear), kernel parameters (kernel scale in RBF and polynomial order in polynomial), and box constraint [71], a hyper-parameter optimization step is included to ensure that hyper-parameters do not significantly affect the evaluation process of the instance selection methods. Manifold methods have been proposed for automatic tuning of hyper-parameters [72, 73]. However, global optimization algorithms are highly recommended because of the non-convex property of hyper-parameter optimization problems. Therefore, in this research, Bayesian optimization as a state-of-the-art method for global optimization of functions is used [74]. Since there are three hyper-parameters, the search space of Bayesian optimization is a 3-dimensional space composed of box constraint, kernel type, and parameter of the selected kernel. The objective function that guides Bayesian optimization towards (near)-optimal hyper-parameters is the average classification loss on validation sets within a stratified q-fold cross-validation scheme (with $q = 4$). In each experimental trial of Bayesian optimization, the training sets are used to train an SVM, while the validation sets are used to calculate the classification loss as the objective function of Bayesian optimization.

Classification loss and preservation rate as two quantitative metrics are calculated according to Eqs. (5) and (6). The reason for concurrently using these two measures is that the ideal objective of instance selection is to minimize the number of instances of a training set (i.e., minimum preservation rate) while maintaining the classification performance (i.e., minimum classification loss).

$$Preservation\ Rate = \frac{number\ of\ instances(Selected\ Data)}{number\ of\ instances(Outer\ Training\ Set)} \times 100 \quad (5)$$

$$Classification\ Loss = \sum_{j=1}^{m} log(1 + exp(-C_j f(\overrightarrow{x_j}))) \quad (6)$$

In Eq. (6), $\overrightarrow{x_j}$ is the jth sample of the test set, f is the decision boundary function obtained by the SVM trained on the selected data, $f(\overrightarrow{x_j})$ is the classification score for $\overrightarrow{x_j}$, and $C_j \in \{-1, +1\}$ is the observed class label, where -1 indicates the negative class (non-building) and +1 indicates the positive class (building) [75].

The parameters of *DR.LSH* significantly control its performance in terms of preservation rate and classification loss. However, it is not initially clear which configuration leads to the more promising results as it depends on the problem at hand. Thus, 60 different combinations of parameters (k, l, and ST) are selected by considering the limit of computational resources and avoiding constructing a myriad of hash functions.

The average classification loss and preservation rate of different configurations are shown in Fig. 7, from which the following important points can be stated:

- The biggest preservation rate is approximately 0.2%, that is, *DR.LSH* could reduce more than 99.8% of instances. Also, it is evident that its corresponding classification loss is insignificant and very close to zero. This proves the great performance of *DR.LSH* in reducing the dataset's records.
- The number of samples eliminated increases with the value of similarity threshold (ST). This is because the parameter ST controls the required degree of similarity for preserving samples.
- Although there is a general inverse relationship between classification loss and preservation rate, classification loss is

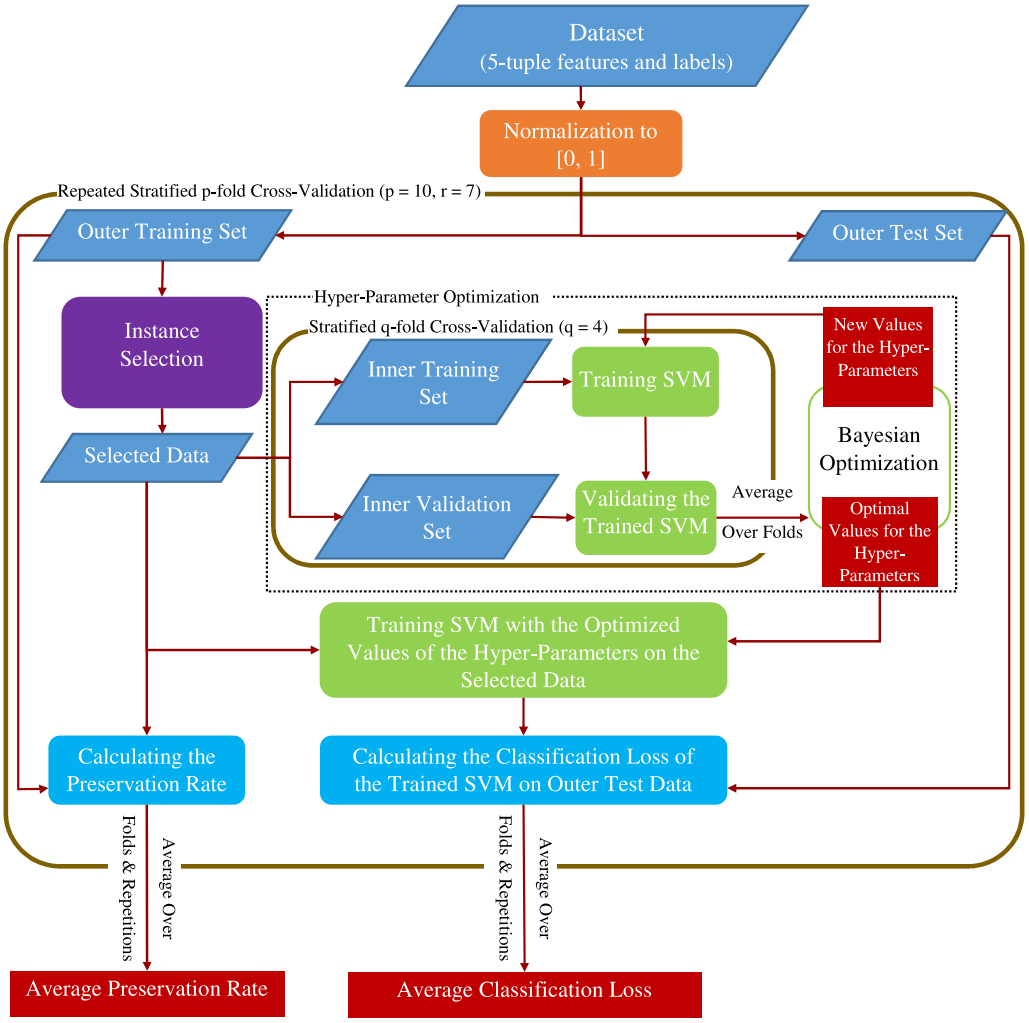

Fig. 6. Workflow for evaluating the performance of the instance selection methods.

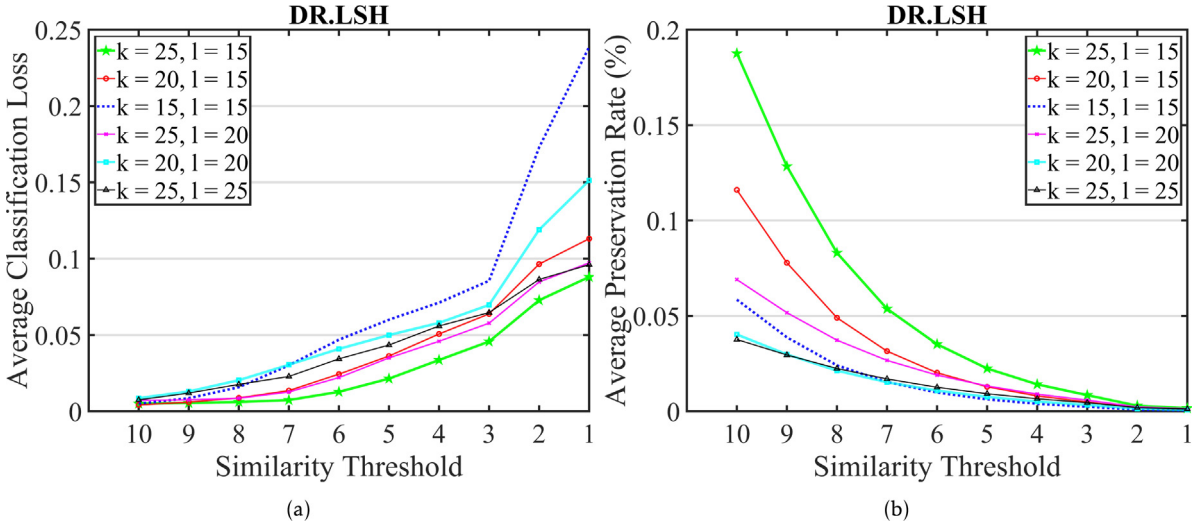

Fig. 7. Quantitative performance measures for different configurations..

almost constant (or has insignificant variations) for $ST = 10$, 9, 8, and 7.

- The configuration of $k = 25$ and $l = 15$ brings about the highest preservation rate. This is because it has the highest value of k and the lowest value of l. A high value of k leads to a high number of buckets with small size in each layer and a low value of l makes $\frac{ST}{l}$ closer to 1 and thus, eliminating condition more strict.

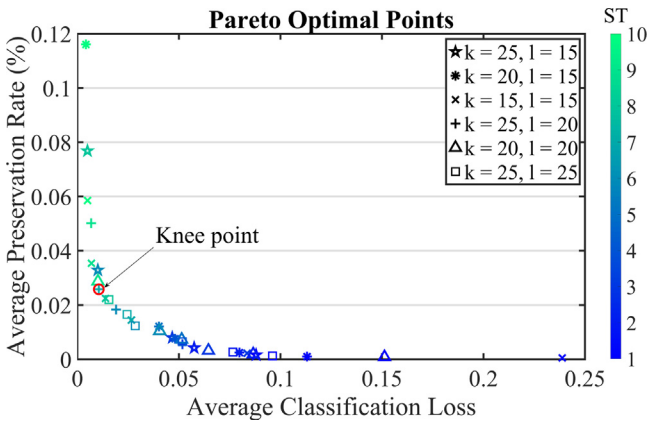

Fig. 8. Pareto optimal configurations by considering both average classification loss and preservation rate. The knee point has the configuration of $k = 25$, $l = 20$, and $ST = 7$.

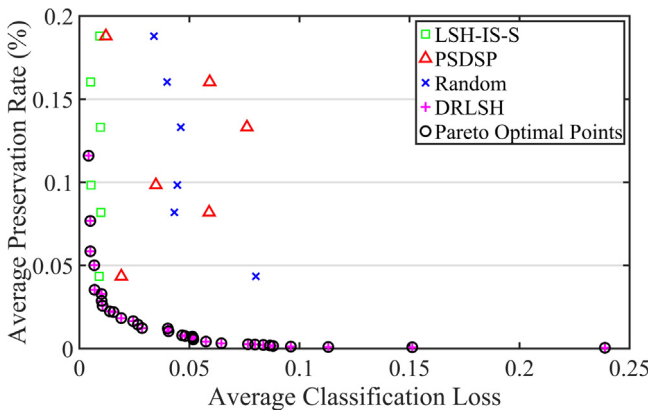

Fig. 9. Comparing the performance of LSH-IS-S, PSDSP, Random, and DR.LSH.

- Reducing *ST* from 10 to 7 is accompanied by a fairly significant drop in the average preservation rate in most of the configurations (especially for $k = 25$ and $l = 15$), but only slight changes in the average classification loss. Thus, $ST = 7$ (or 8) can be considered as the value that strikes a balance between the preservation rate and classification loss.
- Although $ST < 3$ results in very small subsets of training data and consequently faster training phase, these values notably raise the classification loss and cannot be considered as optimal values of *ST*.

Finding the best configuration (*k*, *l*, and *ST*) can be considered as a multi-objective problem since there are two conflict performance criteria (objectives) that should be simultaneously minimized [76]. Unlike single-objective problems, multi-objective problems usually present several optimal solutions that are found by using the Pareto optimality theory [58]. According to the Pareto optimality theory, an optimal solution (configuration) is the one that is not dominated by other feasible solutions (i.e., there exist no other solutions that improve one of the objectives without degrading another objective). A set of all optimal solutions forms the Pareto set which is referred to as the Pareto frontier in the objective space [77].

Among all 60 various configurations, 32 of them are Pareto optimal and are shown in Fig. 8. The non-optimal configurations are not shown in this figure to avoid cluttering. As it is evident, there is a set of optimal alternatives in such a way that an improvement in classification loss can be achieved at the expense of deterioration of the preservation rate and vice versa. Therefore, each point in Fig. 8 can be regarded as the final preferred configuration depending on the importance of each criterion (preservation rate and classification loss).

If there are no additional problem-specific preferences, the most preferred solution can be the one that best balances inherent trade-offs. This type of solution that is referred to as the knee point is well-matched to the point of maximum curvature in the Pareto frontier curve [59]. Due to the discrete nature of the Pareto frontier, there has not been a closed-form and well-defined formulation for finding the point of maximum curvature [78]. In this research, in order to locate the knee point in the Pareto frontier, the L-method is used because of its simplicity and efficiency [79]. The notion of the L-method is that the placement of a knee point has the minimum total root mean square error of the two straight lines fitted to the left and right sides of it. In essence, two straight lines are fitted to the left and right sides of the candidate point. The point that has the minimum total root mean square error is considered as the knee point. Fig. 8 shows the knee point that has the configuration of $k = 25$, $l = 20$, and $ST = 7$. This configuration is also approved by the explanations of Fig. 7.

Table 3 shows the effects of *DR.LSH* on the execution time and classification accuracy of SVM, obtained through the repeated stratified cross-validation scheme. *DR.LSH* could significantly reduce the number of instances and execution time without considerably affecting the original classification accuracy. In other words, the execution time becomes 55 times shorter while the effect of *DR.LSH* on the original classification accuracy is not significant.

The performance of *DR.LSH* is benchmarked against PSDSP, LSH-IS-S, and random selection. Taking advantage of linear time complexity, PSDSP and LSH-IS-S are computationally feasible and appropriate for handling 23,750,000 samples. To make the comparison fair, six various (near)-optimal configurations of PSDSP and LSH-IS-S are selected in such a way that relatively similar preservation rates of *DR.LSH* (less than 0.2%) are obtained.

Fig. 9 compares the performance of the methods in terms of the average preservation rate and classification loss. Since these methods are compared by using two criteria (multi-objective), non-dominated points are selected by Pareto optimality theory. As indicated in Fig. 9, all the Pareto optimal points are from the group of *DR.LSH* and neither LSH-IS-S nor PSDSP nor random could dominate *DR.LSH*. This shows that *DR.LSH* has the best performance in comparison to other methods. Thus, it is sufficiently effective to be used for building extraction in the study area.

The binary map of buildings/non-buildings is extracted by applying the SVM classifier, which is trained on the data selected by *DR.LSH*, to the whole pixels of the study area and utilizing a morphological filter afterward. More precisely, first, *DR.LSH* with the configuration of the knee point ($k = 25$, $l = 20$, and $ST = 7$) is used to select the instances for hyper-parameter optimization and training the SVM. Then, the trained SVM with the optimized hyper-parameters is used to classify the pixels to building and non-building classes based on their input feature values. Finally, a morphological opening with a circular shaped kernel as the structuring element is applied to eliminate the objects whose size is smaller than the smallest building in the row aerial image (removing salt-and-pepper effects). The footprints of buildings extracted are shown in Fig. 10.

Table 4 represents the evaluation results of the output map in terms of completeness, correctness, and quality for both pixel-level and object-level [80,81]. In the object-level measures, if more than 50% of an object is detected, it is considered as a correct detection. As it is clear, having the values of all the

Table 3
Preservation rate, classification accuracy, and execution time.

	Preservation rate (%)	Classification accuracy (%)	Execution time (s)
SVM	100	99.99	26915
DR.LSH (knee point) + SVM	0.026	99.98	492

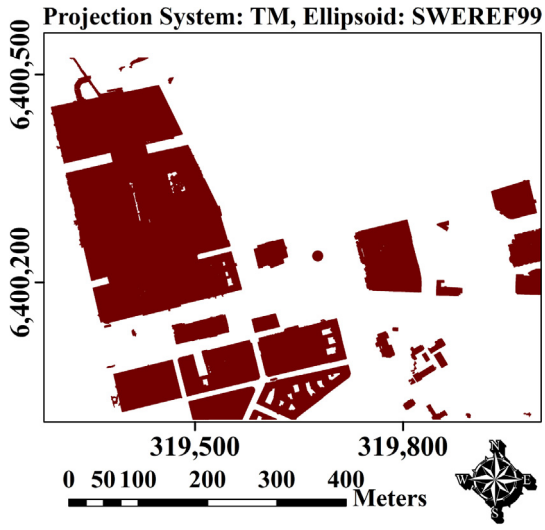

Fig. 10. The buildings extracted by utilizing the trained SVM and a morphological opening.

Table 4
The assessment results of the extracted buildings.

Level	Completeness (%)	Correctness (%)	Quality (%)
Pixel-based	98.6	98.8	97.4
Object-based	93.1	75	71.1

measures greater than 70% indicates that the classifier could be successfully trained on the data selected by *DR.LSH*.

7. Conclusion and future work

It is not always necessary to have a large training set since many of training samples provide no significant contribution to the classification and may not be informative. This is especially a case in pixel-based building extraction from high-resolution aerial images by SVM. Including all pixels of training parts in the training phase gives rise to having a big dataset that contains many unnecessary instances. This is because neighboring pixels in high-resolution aerial images have similar feature values. A smaller training set that is free of similar samples and satisfactorily provides a description of each class is often adequate.

In this paper, an instance selection method (*DR.LSH*) with linear time complexity ($O(n)$) was developed. It is able to select samples (pixels) that convincingly cover the spectral variability and extent of each class in the feature space. The reliance of this method on the property of approximate distance-preserving mapping (i.e., hashing data into buckets), simplicity, and integer-based calculations (low required memory) make it a suitable tool in the big data analysis toolbox. By tuning the input parameters of *DR.LSH* (k, l, and ST), a smaller training dataset that is suitably numerous and effectively represents the extent of each class is acquired.

DR.LSH was evaluated on a huge dataset obtained from the fusion of aerial images and point clouds for building extraction. The dataset consists of 23,750,000 samples with several features such as height and vegetation index. *DR.LSH* was also benchmarked against LSH-IS-S and PSDSP that have linear time complexity as well as a random selection method. Pareto optimality theory was used for a fair comparison of the instance selection methods. The experiment showed that *DR.LSH*, by rapidly selecting a small subset of samples, significantly accelerates the training phase of SVM without substantially impairing the discriminatory power of SVM. Moreover, the results indicated that *DR.LSH* could outperform LSH-IS-S, PSDSP, and the random selection method in terms of preservation rate and classification loss.

Although *DR.LSH* is effective in handling huge datasets, there is still room for improvement. In this paper, only the first sample among similar ones is preserved to make *DR.LSH* fast. Some steps can be added to the algorithm to intelligently select samples without significantly affecting the execution time. For instance, adding some heuristics to select outer samples that are closer to decision boundaries may improve the performance of the algorithm. In addition, *DR.LSH* can be integrated with an analytical approach or an optimization algorithm such as particle swarm optimization [82] and whale optimization [83] algorithms for automatic tuning of the input parameters.

The k-nearest neighbors classifier [84] is one of the most well-known classifiers because of its effectiveness and simplicity. However, this method has high memory and computational complexities with respect to the number of instances. Since *DR.LSH* quickly identifies and removes similar instances, it is a potential instance selection method for the k-nearest neighbors classifier. Thus, further work includes evaluation and/or development of our method to the k-nearest neighbors classifier. Another potential research line is to adapt *DR.LSH* to prototype generation as a promising research line that was not considered in this paper. This idea can be developed by analyzing the structure of buckets obtained by LSH.

CRediT authorship contribution statement

Mohammad Aslani: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft. **Stefan Seipel:** Conceptualization, Writing - review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partly funded by the European Regional Development Fund (ERDF), contract ID 20201871. The authors would like to appreciate Anders Ekholm and Lantmäteriet, the Swedish mapping, cadastral and land registration authority, for providing the data for this study.

Appendix. Proof of linearity of DR.LSH's time complexity

The total computational cost of *DR.LSH* consists of the time complexity for 1- hashing instances into buckets (lines 2–7 of

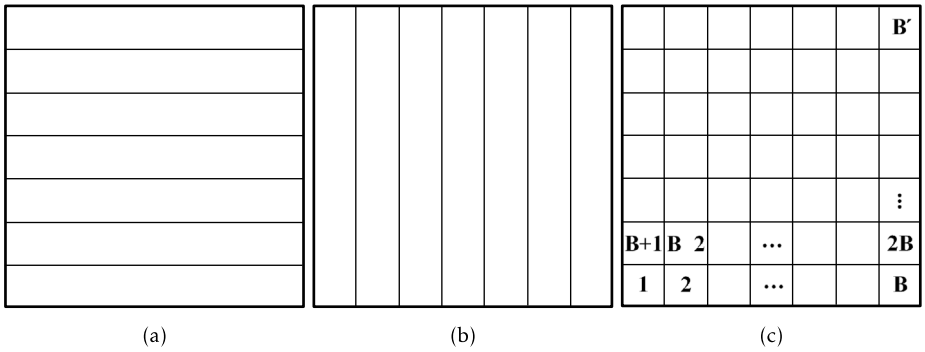

Fig. A.1. (a) and (b) two layers of buckets. (c) the buckets obtained by intersecting (a) and (b).

the algorithm) and 2- removing similar instances of each class based on the similarity index (lines 8–26). In what follows, the time complexity of each part is separately explained.

A.1. Time complexity of lines 2–7 in DR.LSH algorithm

Since each instance is projected from the feature space into a new space by using k hash functions in l layers, the time complexity is $O(nkl)$ where n is the number of instances.

A.2. Time complexity of lines 8–26 in DR.LSH algorithm

To simplify the proof of time complexity, it is assumed that 1- instances of each class are evenly distributed in each bucket, 2- buckets have regular shape, and 3- each layer has the same number of buckets. The upper bound of the time complexity of the second part happens when $ST = l$. In this case, the number of iterations of lines 11–23 of the algorithm is equal to the total number of buckets obtained by intersecting l layers of buckets (rather than the number of instances). This is because only one instance in each bucket, obtained by intersecting the layers, is preserved and other instances that share the same bucket are removed and will not be checked in next iterations.

Fig. A.1 shows an example of intersecting two layers of buckets in the feature space. B is the number of buckets in each layer and B' is the total number of buckets produced by intersecting the layers. It should be noted that instances are not shown in this figure to avoid cluttering. If it is assumed that *DR.LSH* goes through instances from left to right and down to up (for the sake of simplicity), the total number of instances of class C retrieved and processed in the iterations of lines 11–23 is calculated as follows (n_C is the number of instances in class C):

$$
\begin{aligned}
& np_1 = 2\frac{n_c}{B} \\
& np_2 = 2\frac{n_c}{B} - \frac{n_c}{B'}(1+0) \\
& np_3 = 2\frac{n_c}{B} - \frac{n_c}{B'}(2+0) \\
& \quad \vdots \\
+ \quad & np_{B'} = 2\frac{n_c}{B} - \frac{n_c}{B'}(B-1+B-1)
\end{aligned}
$$

$$
np_T = 2\frac{n_c}{B}B' - \frac{n_c}{B'}\left(\sum_{i=0}^{B-1}\sum_{j=0}^{B-1} i+j\right) = 2\frac{n_c}{B}B' - \frac{n_c}{B'}\left(B^2(B-1)\right) \quad \text{(A.1)}
$$

The general form of the above equation for l layers is:

$$
\begin{aligned}
np_T &\simeq l\frac{n_c}{B}B' - \frac{n_c}{B'}\left(\sum_{i_1=0}^{B-1}\sum_{i_2=0}^{B-1}\cdots\sum_{i_l=0}^{B-1} i_1+i_2+\cdots+i_l\right) \\
&\simeq l\frac{n_c}{B}B' - \frac{n_c}{B'}\left(\frac{B^l(B-1)l}{2}\right) \\
&\simeq n_c l\left(\frac{B'}{B} - \frac{B^l(B-1)}{2B'}\right)
\end{aligned} \quad \text{(A.2)}
$$

The total number of instances retrieved and processed for all classes is:

$$
Np_T \simeq nl\left(\frac{B'}{B} - \frac{B^l(B-1)}{2B'}\right) \quad \text{(A.3)}
$$

Thus, the upper bound of the time complexity of the second part is $O\left(nl\left(\frac{B'}{B} - \frac{B^l(B-1)}{2B'}\right)\right)$.

A.3. Total time complexity of DR.LSH algorithm

The upper bound of *DR.LSH*'s time complexity is $O\left(nl\left(\frac{B'}{B} - \frac{B^l(B-1)}{2B'}\right) + nkl\right)$. As it is clear, it is linear with respect to n.

References

[1] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[2] G.M. Foody, The effect of mis-labeled training data on the accuracy of supervised image classification by SVM, in: 2015 IEEE International Geoscience and Remote Sensing Symposium, IGARSS, 2015, pp. 4987–4990.

[3] L. Liu, W. Huang, C. Wang, Hyperspectral image classification with kernel-based least-squares support vector machines in sum space, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 11 (4) (2018) 1144–1157.

[4] U. Maulik, D. Chakraborty, Remote Sensing Image Classification: A survey of support-vector-machine-based advanced techniques, IEEE Geosci. Remote Sens. Mag. 5 (1) (2017) 33–52.

[5] V.N. Vapnik, Statistical Learning Theory, John Wiley and Sons, New York, 1998, p. 768.

[6] G.M. Foody, Status of land cover classification accuracy assessment, Remote Sens. Environ. 80 (1) (2002) 185–201.

[7] B. Demir, L. Minello, L. Bruzzone, Definition of effective training sets for supervised classification of remote sensing images by a novel cost-sensitive active learning method, IEEE Trans. Geosci. Remote Sens. 52 (2) (2014) 1272–1284.

[8] G.M. Foody, A. Mathur, Toward intelligent training of supervised image classifications: directing training data acquisition for SVM classification, Remote Sens. Environ. 93 (1) (2004) 107–117.

[9] G.M. Foody, A. Mathur, C. Sanchez-Hernandez, D.S. Boyd, Training set size requirements for the classification of a specific class, Remote Sens. Environ. 104 (1) (2006) 1–14.

[10] F. Chang, C. Guo, X. Lin, C. Liu, C. Lu, Tree decomposition for large-scale SVM problems, in: 2010 International Conference on Technologies and Applications of Artificial Intelligence, IEEE, Hsinchu, Taiwan, 2010, pp. 233–240.

[11] S. Abe, Support vector machines for pattern classification, in: S. Singh (Ed.), Advances in Computer Vision and Pattern Recognition, Springer, London, 2010.
[12] L. Kaufmann, Solving the quadratic programming problem arising in support vector classification, in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, 1999, pp. 147–168.
[13] D. Pavlov, J. Mao, B. Dom, Scaling-up support vector machines using boosting algorithm, in: Proceedings 15th International Conference on Pattern Recognition, Vol. 2, ICPR-2000, IEEE, Barcelona, 2000, pp. 219–222.
[14] J. Platt, Fast training of support vector machines using sequential minimal optimization, in: B. Schölkopf, C.J.C. Burges, A.J. Smola (Eds.), Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, 1999, pp. 185–208.
[15] Á. Arnaiz-González, J.-F. Díez-Pastor, J.J. Rodríguez, C. García-Osorio, Local sets for multi-label instance selection, Appl. Soft Comput. 68 (2018) 651–666.
[16] R.M. Marone, F. Camara, S. Ndiaye, LSIS: Large scale instance selection algorithm for big data, in: 3rd IEEE International Conference on Computer and Communications, ICCC, IEEE, Chengdu, China, 2017, pp. 2353–2356.
[17] L. Guo, S. Boukir, Fast data selection for SVM training using ensemble margin, Pattern Recognit. Lett. 51 (2015) 112–119.
[18] C. Liu, W. Wang, M. Wang, F. Lv, M. Konan, An efficient instance selection algorithm to reconstruct training set for support vector machine, Knowl.-Based Syst. 116 (2017) 58–73.
[19] J. Nalepa, M. Kawulok, Selecting training sets for support vector machines: a review, Artif. Intell. Rev. 52 (2) (2018) 857–900.
[20] S. García, J. Luengo, F. Herrera, in: J. Kacprzyk, L.C. Jain (Eds.), Data Preprocessing in Data Mining, Vol. 72, Springer, Heidelberg, Berlin, 2015, p. 320.
[21] K. Schmidt, T. Behrens, T. Scholten, Instance selection and classification tree analysis for large spatial datasets in digital soil mapping, Geoderma 146 (1) (2008) 138–146.
[22] J. Cervantes, F. García Lamont, A. López-Chau, L. Rodríguez Mazahua, J. Sergio Ruíz, Data selection based on decision tree for SVM classification on large data sets, Appl. Soft Comput. 37 (2015) 787–798.
[23] J.A. Olvera-López, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad, J. Kittler, A review of instance selection methods, Artif. Intell. Rev. 34 (2) (2010) 133–143.
[24] S. Garcia, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: Taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435.
[25] J. Derrac, S. García, F. Herrera, A survey on evolutionary instance selection and generation, Int. J. Appl. Metaheuristic Comput. 1 (1) (2010) 60–92.
[26] Y. Lee, S. Huang, Reduced support vector machines: A statistical theory, IEEE Trans. Neural Netw. 18 (1) (2007) 1–13.
[27] J. Wang, P. Neskovic, L.N. Cooper, Selecting data for fast support vector machines training, in: K. Chen, L. Wang (Eds.), Trends in Neural Computation, Springer, Berlin, Heidelberg, 2007, pp. 61–84.
[28] X. Li, J. Cervantes, W. Yu, Fast classification for large data sets via random selection clustering and Support Vector Machines, Intell. Data Anal. 16 (6) (2012) 897–914.
[29] A. Lyhyaoui, M. Martinez, I. Mora, M. Vaquez, J.-L. Sancho, A.R. Figueiras-Vidal, Sample selection via clustering to construct support vector-like classifiers, IEEE Trans. Neural Netw. 10 (6) (1999) 1474–1481.
[30] M.B. de Almeida, A.d.P. Braga, J.P. Braga, SVM-KM: speeding SVMs learning with a priori cluster selection and k-means, in: Proceedings. Vol. 1. Sixth Brazilian Symposium on Neural Networks, RJ, Brazil, 2000, pp. 162–167.
[31] R. Koggalage, S. Halgamuge, Reducing the number of training samples for Fast Support Vector Machine Classification, Neural Inf. Process.-Lett. Rev. 2 (2004) 57–65.
[32] X.-J. Shen, L. Mu, Z. Li, H.-X. Wu, J.-P. Gou, X. Chen, Large-scale support vector machine classification with redundant data reduction, Neurocomputing 172 (2016) 189–197.
[33] J. Cervantes, X. Li, W. Yu, in: A. Gelbukh, C.A. Reyes-Garcia (Eds.), Support Vector Machine Classification Based on Fuzzy Clustering for Large Data Sets BT - MICAI 2006: Advances in Artificial Intelligence, Springer, Berlin, Heidelberg, 2006, pp. 572–582.
[34] D. Boley, D. Cao, Training support vector machine using adaptive clustering, in: Proceedings of the Fourth SIAM International Conference on Data Mining, Florida, USA, 2004.
[35] J. Cervantes, X. Li, W. Yu, K. Li, Support vector machine classification for large data sets via minimum enclosing ball clustering, Neurocomputing 71 (4) (2008) 611–619.
[36] H. Yu, J. Yang, J. Han, Classifying large data sets using SVMs with hierarchical clusters, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Washington, 2003, pp. 306–315.
[37] D. Wang, L. Shi, Selecting valuable training samples for SVMs via data structure analysis, Neurocomputing 71 (13) (2008) 2772–2781.
[38] Z.-Q. Zeng, H.-R. Xu, Y.-Q. Xie, J. Gao, A geometric approach to train SVM on very large data sets, in: 3rd International Conference on Intelligent System and Knowledge Engineering, Vol. 1, IEEE, Xiamen, 2008, pp. 991–996.
[39] J.A. Olvera-López, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad, A new fast prototype selection method based on clustering, Pattern Anal. Appl. 13 (2) (2010) 131–141.
[40] A. López Chau, X. Li, W. Yu, Convex and concave hulls for classification with support vector machine, Neurocomputing 122 (2013) 198–209.
[41] H. Shin, S. Cho, Neighborhood property-based pattern selection for support vector machines, Neural Comput. 19 (3) (2007) 816–855.
[42] S. Abe, T. Inoue, Fast training of support vector machines by extracting boundary data, in: G. Dorffner, H. Bischof, K. Hornik (Eds.), International Conference on Artificial Neural Networks, Springer, Berlin, Heidelberg, 2001, pp. 308–313.
[43] Y.-G. Liu, Q. Chen, R.-Z. Yu, Extract candidates of support vector from training set, in: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, Vol. 5, IEEE, Xi'an, China, 2003, pp. 3199–3202.
[44] J. Carbonera, M. Abel, A density-based approach for instance selection, in: 27th International Conference on Tools with Artificial Intelligence, ICTAI, IEEE, Vietri sul Mare, Italy, 2015, pp. 768–774.
[45] H. Shin, S. Cho, Pattern selection for support vector classifiers, in: H. Yin, N. Allinson, R. Freeman, J. Keane, S. Hubbard (Eds.), International Conference on Intelligent Data Engineering and Automated Learning, Springer, Berlin, Heidelberg, 2002, pp. 469–474.
[46] R. Wang, S. Kwong, Sample selection based on maximum entropy for support vector machines, in: 2010 International Conference on Machine Learning and Cybernetics, Vol. 3, IEEE, Qingdao, China, 2010, pp. 1390–1395.
[47] Y. Li, L. Maguire, Selecting critical patterns based on local geometrical and statistical information, IEEE Trans. Pattern Anal. Mach. Intell. 33 (6) (2011) 1189–1201.
[48] Z. Zhu, Z. Wang, D. Li, W. Du, NearCount: Selecting critical instances based on the cited counts of nearest neighbors, Knowl.-Based Syst. 190 (2020) 105196.
[49] L. Guo, S. Boukir, N. Chehata, Support vectors selection for supervised learning using an ensemble approach, in: 20th International Conference on Pattern Recognition, IEEE, Istanbul, Turkey, 2010, pp. 37–40.
[50] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[51] R.O. Leo Breiman, Jerome Friedman, Charles J. Stone, Classification and Regression Trees, Chapman and Hall, London, 1984, p. 368.
[52] L. Guo, S. Boukir, Margin-based ordered aggregation for ensemble pruning, Pattern Recognit. Lett. 34 (6) (2013) 603–609.
[53] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[54] J.L. Carbonera, M. Abel, Efficient instance selection based on spatial abstraction, in: 30th International Conference on Tools with Artificial Intelligence, ICTAI, IEEE, Volos, Greece, 2018, pp. 286–292.
[55] Á. Arnaiz-González, J.-F. Díez-Pastor, J.J. Rodríguez, C. García-Osorio, Instance selection of linear complexity for big data, Knowl.-Based Syst. 107 (2016) 83–95.
[56] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: VLDB '99 Proceedings of the 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, 1999, pp. 518–529.
[57] J. Wang, W. Liu, S. Kumar, S. Chang, Learning to hash for indexing big data-a survey, Proc. IEEE 104 (1) (2016) 34–57.
[58] M. Ehrgott, Multicriteria Optimization, Springer, Berlin Heidelberg, 2005, p. 323.
[59] M. Antunes, D. Gomes, R. Aguiar, Knee/elbow estimation based on first derivative threshold, in: 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications, BigDataService, Bamberg, Germany, 2018, pp. 237–240.
[60] M. Datar, N. Immorlica, P. Indyk, V. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the Twentieth Annual Symposium on Computational Geometry, ACM, New York, NY, USA, 2004, pp. 253–262.
[61] M. Gerke, J. Xiao, Fusion of airborne laserscanning point clouds and images for supervised and unsupervised scene classification, ISPRS J. Photogramm. Remote Sens. 87 (2014) 78–92.
[62] M. Awrangjeb, M. Ravanbakhsh, C.S. Fraser, Automatic detection of residential buildings using LIDAR data and multispectral imagery, ISPRS J. Photogramm. Remote Sens. 65 (5) (2010) 457–467.
[63] X. Gao, M. Wang, Y. Yang, G. Li, Building extraction from RGB VHR images using shifted shadow algorithm, IEEE Access 6 (2018) 22034–22045.
[64] X. Huang, L. Zhang, An SVM ensemble approach combining spectral, structural, and semantic features for the classification of high-resolution remotely sensed imagery, IEEE Trans. Geosci. Remote Sens. 51 (1) (2013) 257–272.
[65] M. Turker, D. Koc-San, Building extraction from high-resolution optical spaceborne images using the integration of support vector machine (SVM) classification, Hough transformation and perceptual grouping, Int. J. Appl. Earth Obs. Geoinf. 34 (2015) 58–69.

[66] K. Zhang, S.-C. Chen, D. Whitman, M.-L. Shyu, J. Yan, C. Zhang, A progressive morphological filter for removing nonground measurements from airborne LIDAR data, IEEE Trans. Geosci. Remote Sens. 41 (4) (2003) 872–882.
[67] R.C. Gonzalez, R.E. Woods, Digital Image Processing, second ed., Prentice Hall, Upper Saddle River, New Jersey, USA, 2008.
[68] S. Riley, S. Degloria, S.D. Elliot, A terrain ruggedness index that quantifies topographic heterogeneity, Int. J. Sci. 5 (1999) 23–27.
[69] H. Ishibuchi, Y. Nojima, Repeated double cross-validation for choosing a single solution in evolutionary multi-objective fuzzy classifier design, Knowl.-Based Syst. 54 (2013) 22–31.
[70] G. Jiang, W. Wang, Error estimation based on variance analysis of k-fold cross-validation, Pattern Recognit. 69 (2017) 94–106.
[71] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, R.C. Williamson, Estimating support of a high-dimensional distribution, Neural Comput. 13 (7) (2001) 1443–1471.
[72] M. Feurer, F. Hutter, Hyperparameter optimization, in: F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges, Springer, 2019, pp. 3–33.
[73] S. Wang, J. Yu, E. Lapira, J. Lee, A modified support vector data description based novelty detection approach for machinery components, Appl. Soft Comput. 13 (2) (2013) 1193–1205.
[74] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, N. de Freitas, Taking the human out of the loop: A review of Bayesian optimization, Proc. IEEE 104 (1) (2016) 148–175.
[75] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning. Data Mining, Inference, and Prediction, Springer, New York, 2009.
[76] D. Roijers, S. Whiteson, Multi-Objective Decision Making, in: Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 11, 2017, pp. 1–129.
[77] A. Keller, Multi-Objective Optimization in Theory and Practice I: Classical Methods, Bentham Science Publishers, Sharjah, 2017.
[78] V. Satopaa, J. Albrecht, D. Irwin, B. Raghavan, Finding a "kneedle" in a haystack: Detecting knee points in system behavior, in: 2011 31st International Conference on Distributed Computing Systems Workshops, IEEE, Minneapolis, MN, USA, 2011, pp. 166–171.
[79] S. Salvador, P. Chan, Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms, in: 16th IEEE International Conference on Tools with Artificial Intelligence, IEEE, Boca Raton, FL, USA, 2004, pp. 576–584.
[80] L. Cai, W. Shi, Z. Miao, M. Hao, Accuracy assessment measures for object extraction from remote sensing images, Remote Sens. 10 (2) (2018) 303.
[81] M. Rutzinger, F. Rottensteiner, N. Pfeifer, A comparison of evaluation techniques for building extraction from airborne laser scanning, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2 (1) (2009) 11–20.
[82] M. Clerc, From theory to practice in particle swarm optimization, in: B.K. Panigrahi, Y. Shi, M.-H. Lim (Eds.), Handbook of Swarm Intelligence: Concepts, Principles and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 3–36.
[83] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67.
[84] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inform. Theory 13 (1) (1967) 21–27.