Applied Soft Computing 47 (2016) 515–533

Contents lists available at ScienceDirect

Applied Soft Computing

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

# A convex hull-based data selection method for data driven models

H.R. Khosravani[a,b], A.E. Ruano[a,b,*], P.M. Ferreira[c]

[a] IDMEC, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[b] University of Algarve, Portugal
[c] LaSIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal

ARTICLE INFO

*Article history:*
Received 17 February 2016
Received in revised form 9 May 2016
Accepted 11 June 2016
Available online 21 June 2016

*Keywords:*
Convex hull
Data selection problem
Classification
Regression
Neural networks
Support vector machines

ABSTRACT

The accuracy of classification and regression tasks based on data driven models, such as Neural Networks or Support Vector Machines, relies to a good extent on selecting proper data for designing these models, covering the whole input range in which they will be employed. The convex hull algorithm can be applied as a method for data selection; however the use of conventional implementations of this method in high dimensions, due to its high complexity, is not feasible. In this paper, we propose a randomized approximation convex hull algorithm which can be used for high dimensions in an acceptable execution time, and with low memory requirements. Simulation results show that data selection by the proposed algorithm (coined as ApproxHull) can improve the performance of classification and regression models, in comparison with random data selection.

## 1. Introduction

In many machine learning and data mining problems two basic tasks have to be considered: feature selection and instance selection. The former denotes choosing a subset of all available features so that the selected subset has the strongest relation to the model output and yields improved model performance. The latter refers to sample selection where we are interested in selecting a subset of useful and informative data samples (denoted $S$) among all existing data samples (denoted $T$). The goal is that the model obtained by using $S$ can maintain or even exceed the performance level (for instance, accuracy) that would be attained by using $T$. The instance selection process not only helps decreasing the run time of the training process but also has the benefit of reducing the memory requirements of learning algorithms. This is important when classification or regression tasks rely on existing large-size training sets.

Generally speaking, instance selection methods can be classified from the search direction and selection criterion points of view. Regarding the search direction, the methods are categorized as incremental or decremental. In the former, the selection process starts with $S = \emptyset$ and progresses iteratively by inserting selected samples from $T$ into $S$. In the latter, in contrast, the selection process starts with$S = T$ and superfluous samples are discarded from $S$ in an iterative manner. Finally, for both methods, we have that $S \subset T$ at the end of the selection process [1,2]. From the selection criterion point of view, instance selection methods are classified as wrapper or filter methods.

Wrapper methods use a model as a selection criterion, where the performance of the model is evaluated based on a subset of samples iteration by iteration to select those samples which have the most contribution on the model accuracy. Most works found in literature on wrapper methods relate to classification tasks. Collectively they may be further subcategorized into two groups. The first gathers methods which are based on $k - NN$ ($K$ Nearest Neighbors) classifiers, whereas the second group involves a broad class of wrapper methods that can be based on any classifier. In the group of $k - NN$ based methods, the CNN (Condensed Nearest Neighbor rule) is one of the earliest incremental methods which was proposed by Hart [3]. It focuses on misclassified samples as critical samples that matter the most to the $k - NN$ classifier to ensure that unlabeled samples which are similar to the misclassified ones are correctly classified [4]. To

* Corresponding author at: University of Algarve, Portugal.
*E-mail addresses:* hkhosravani@csi.fct.ualg.pt (H.R. Khosravani), aruano@ualg.pt, (A.E. Ruano), pmf@ciencias.ulisboa.pt (P.M. Ferreira).

http://dx.doi.org/10.1016/j.asoc.2016.06.014
1568-4946/© 2016 Elsevier B.V. All rights reserved.

improve the CNN method, the SNN (Selective Nearest Neighbor rule) and the GCNN (Generalized Condensed Nearest Neighbor rule) were proposed by Ritter et al. [5] and Chou et al. [6], respectively. Besides considering critical samples, some noteworthy $k-NN$ rule based methods are focused on the removal of noisy samples: the ENN (Edited Nearest Neighbor rule) [7], the $allk-NN$ [8], a family of five incremental methods coined DROP1 to DROP5 (Decremental Reduction Optimization Procedure, 1–5) [2], and the ICF (Iterative Case Filtering) [9]. Regarding general wrapper methods, arbitrary classifiers can be used as the criterion for instance selection. Search algorithms have had a significant contribution, from which evolutionary algorithms have received most attention in the literature of instance selection. Specifically, the application of Genetic Algorithms (GA) can be found in [10–13]. Besides evolutionary algorithms, other search algorithms such as tabu search, BSE (Backward Sequential Edition) and RFOS (Restricted Floating Object Selection) have also been considered [14–17].

Unlike wrapper methods, filter methods employ a model independent selection function to choose informative samples [1]. This means that the accuracy of the model does not have any contribution in the selection criterion; instead, a selection rule is applied. One group of filter methods applied in classification problems focuses on selecting class border samples [18,19]. Alternatively some authors employed clustering methods in order to select instances that can be considered representatives for a cluster of instances [20–25].

Neural networks and Support Vector Machines (SVM) are two examples of well-established data driven machine learning approaches for classification and regression tasks. Since the models generated by these approaches are data driven, data must be selected in such way that it covers the whole input range in which the model is to be employed. To achieve this goal two different methods using Principal Components Analysis (PCA) and convex hull have been proposed [26,27]. Since SVM training has high a learning time, having a large training dataset for online classification is infeasible as the existing model should be retrained with newly arriving samples along with an already large training dataset. In this condition, a reasonable portion of the large training dataset should be selected for retraining the existing model. For this purpose, methods for data reduction in SVM classification models have been proposed [28,29], including methods based on the convex hull [27,30].

The identification of convex hull vertices is a time consuming task, as the complexity of real convex hull algorithms in high dimensions is $O\left(n^{\lfloor \frac{d}{2} \rfloor}\right)$ [31] ($n$ and $d$ denoting the number of samples and sample dimension respectively). In this paper we propose a Randomized Approximation convex hull algorithm, coined ApproxHull, to overcome both the high execution time and memory requirements, resulting from the curse of dimensionality associated with the convex hull algorithm complexity. ApproxHull can be used not only for off-line training, but also for online model adaptation.

The rest of the paper is organized as follows: Section 2 gives a brief description on existing convex hull algorithms. Section 3 addresses the proposed algorithm for determining an approximation of the convex hull in high dimensions. Simulation results are presented in Section 4. The run time analysis of ApproxHull is addressed in Section 5. The comparison of the proposed algorithm with Quickhull [32] from memory requirements point of view is described in Section 6. Conclusions and future work are addressed in Section 7.

## 2. Definitions and related work

### 2.1. Convex hull definition

From a computational geometry's point of view, an object in Euclidean space is convex if for every pair of points within the object, every point on the straight line segment that joins them is also within the object. A set $S$ is convex if, for every pair, $u, v \in S$, and all $t \in [0, 1]$, the point $(1-t)u + tv$ is in $S$. Moreover, if $S$ is a convex set, for any $u_1, u_2, \ldots, u_r \in S$, and any nonnegative numbers $\left\{\lambda_1, \lambda_2, \ldots, \lambda_r\right\} : \sum_{i=1}^{r} \lambda_i = 1$, the vector $\sum_{i=1}^{r} \lambda_i u_i$ is called a convex combination of $u_1, u_2, \ldots, u_r$. According to the definitions above, the convex hull or convex envelope of set $X$of points in the Euclidean space can be defined in terms of convex sets or convex combinations:

- the minimal convex set containing $X$, or
- the intersection of all convex sets containing $X$, or
- the set of all convex combinations of points in $X$ .

Based on the definition of convex hull, a $k$ -simplex is a $k$-dimensional polytope which is the convex hull of $k+1$ affinely independent points. Intuitively, 0-simplex, 1-simplex, 2-simplex and 3-simplex correspond to a point, a line segment, a triangle and a tetrahedron, respectively. Generally, a $k$-simplex consists of the elements called $i$-faces where $i \leq k-1$. 0-faces, 1-faces and $(k-1)$-faces are called vertices, edges and facets of the $k$ -simplex, respectively.

### 2.2. Convex hull algorithms

Convex hull algorithms can be categorized from three points of view. An algorithm can be deterministic or randomized depending on the order of vertices found. If the order is fixed from run to run, the algorithm is deterministic [33]; otherwise, it is randomized [34]. Furthermore, an algorithm can be considered as a real or approximation algorithm. If it is capable of identifying all vertices of the real convex hull, the algorithm is coined as real [32]; otherwise, it is considered an approximation algorithm [35,36]. Finally, we can also classify convex hull algorithms into offline and online algorithms. The former uses all the data to compute the convex hull, while the latter employ newly arrived points to adapt an already existing convex hull [31].

Although many algorithms have been proposed for identifying the convex hull of datasets in low dimensions, still there is no efficient algorithm available to find the convex hull in higher dimensions. The complexity of the majority of proposed algorithms for two or three dimensions is $O(nlogn)$ while for high dimensions the complexity is $O\left(n^{\lfloor \frac{d}{2} \rfloor}\right)$, where $n$ is the number of samples in dataset and $d$ is the sample dimension. According to the upper bound theory in computational geometry [37], the maximum number of facets for a convex hull with $m$ vertices is $O\left(m^{\lfloor \frac{d}{2} \rfloor}\right)$, which reflects the large memory requirements for those algorithms that construct the convex hull by enumerating facets, e.g., randomized incremental algorithms [34] and Quickhull [32].

Among all proposed algorithms, Quickhull is considered a quick deterministic real convex hull algorithm which is faster than other proposed algorithms in low dimensions. For dimensions $d \leq 3$ Quickhull runs in time $O(n \log r)$, where $n$ and $r$are the number of points in the underlying dataset and the number of processed points, respectively. For $d \geq 4$, Quickhull runs in time $O\left(nf_r/r\right)$, where $f_r$ is the maximum number of facets for $r$ vertices. Since $f_r = O\left(r^{\lfloor \frac{d}{2} \rfloor} / \lfloor \frac{d}{2} \rfloor !\right)$, for high dimensions a massive number of facets