Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-10 (2015) 123–128

A Randomized Approximation Convex Hull Algorithm for High Dimensions

Antonio Ruano*. Hamid Reza Khosravani. Pedro M. Ferreira*****

**University Of Algarve 8005-139 Faro, Portugal (e-mail: aruano@ ualg.pt).*
*** University Of Algarve 8005-139 Faro, Portugal (e-mail:hkhosravani@csi.fct.ualg.pt)*
**** LaSIGE, Faculty of Sciences, University of Lisbon, Portugal(e-mail: pmf@ciencias.ulisboa.pt)*

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 that covers the whole input ranges in which they will be employed. The convex hull algorithm is 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.

Keywords: Convex Hull, Data Selection Problem, Classification, Regression, Neural Networks, Support Vector Machines.

1. INTRODUCTION

Neural networks and Support Vector Machines (SVM), as well as other data driven machine learning approaches, are well established methods for classification and regression tasks. Since the models generated by these approaches are data driven, selecting suitable data from large datasets for the design phase is a crucial task, as the accuracy of these models is affected by the data in the training dataset. Data must be selected in such way that it covers the whole input ranges in which the model is to be employed. To achieve this goal, (Malosek and Stopjakova, 2006, Wang et al., 2013) presented two different methods using Principal Components Analysis (PCA) and convex hull. In an on-line learning context, the convex hull was applied for sample reduction in classification and regression, where the existing model should be retrained with newly arriving samples along with a reasonable portion of the current training dataset (Wang et al., 2013, Lopez Chau et al., 2013).

The identification of the convex hull vertices is a time consuming task, as the complexity of real convex hull algorithms in high dimensions is $O(n^{\lfloor \frac{d}{2} \rfloor})$ (Bayer, 1999), where n and d denote the number of samples and sample dimension respectively. In this paper, we propose a Randomized Approximation Convex Hull Algorithm to overcome both the high execution time and the memory requirements, which result from the convex hull algorithm complexity for high-dimensional data. The proposed algorithm 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 our proposed algorithm for determining an approximation of the convex hull in high dimensions. Section 4 presents simulation results. Conclusions are presented in Section 5.

2. RELATED WORKS

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 $u_1, u_2, \dots, u_r \in S$, and any nonnegative numbers $\{\lambda_1, \lambda_2, \dots, \lambda_r\}$: $\sum_{i=1}^{r} \lambda_i = 1$, the vector $\sum_{i=1}^{r} \lambda_i u_i$ is called a convex combination of $u_1, u_2, \dots, u_r$. 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.

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 (Graham, 1972); otherwise, it is randomized (Clarkson and Shor, 1989). 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 real (Barber et al., 1996); otherwise, it is considered an approximation

[1] This work was supported by QREN SIIDT 38798, and IDMEC, under LAETA

10.1016/j.ifacol.2015.08.119

(Bentley et al., 1982, Khosravani et al., 2013). 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 (Bayer, 1999).

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 time complexity of the majority of proposed algorithms for two or three dimensions is $O(n\,log\,n)$ while for high dimensions, the complexity is $O(n^{\lfloor \frac{d}{2} \rfloor})$, where *n* is the number of samples in dataset and *d* is the sample dimension. According to the upper bound theory in computational geometry (Seidel, 1995), the maximum number of facets for a convex hull with m vertices is $O(m^{\lfloor \frac{d}{2} \rfloor})$, which reflects the large memory requirements for those algorithms that construct the convex hull by enumerating facets, e.g., the randomized incremental algorithm (Clarkson and Shor, 1989) and the Quickhull (Barber et al., 1996).

Among all proposed algorithms, Quickhull is considered as 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 $O(nf_r/r)$, where f_r is the maximum number of facets for r vertices. Since $f_r = O(r^{\lfloor \frac{d}{2} \rfloor} / \lfloor \frac{d}{2} \rfloor !)$, for high dimensions a massive number of facets would be generated for r vertices. Consequently Quickhull is not feasible for high dimensions, both in terms of execution time and memory requirements, e.g., for $d > 8$ it suffers from insufficient memory problems.

Very recently, an on-line algorithm (Wang et al., 2013) has been proposed for application to SVMs. Its time complexity is at most $O(nd^4)$, which means that, for problems with $d > 8$, it has smaller complexity than the existing techniques. It incrementally forms an approximated convex hull of a dataset on the basis of two thresholds, *L* and *M*. The algorithm starts from a *d*-simplex and ends with an approximated convex hull with at most *M* vertices. Since a *d*-simplex has *d*+1 facets, it divides the space into *d*+1 partitions. In the first step, each partition whose number of samples is greater than L is divided into d new partitions, based on the furthest sample to the corresponding facet of each partition. This task is performed repeatedly for the new generated partitions and the furthest samples are marked as convex hull vertices. Afterwards, the sample whose distance to the current generated convex hull is maximum is marked as a vertex of convex hull. The procedure is executed until the number of vertices reaches the threshold *M*. Although the algorithm proposed in (Wang et al., 2013) is feasible to execute in high dimensions, it incorporates vertices which do not belong to the set of vertices of the real convex hull, as will be demonstrated in the results.

3. PROPOSED ALGORITHM

In order to overcome the shortcomings of Quickhull and the algorithm proposed in (Wang et al., 2013), we propose a randomized approximation algorithm so that on one hand, it treats memory complexity efficiently and on the other hand, it identifies the vertices which are exactly the vertices of the real convex hull. Moreover, this algorithm is capable to be applied in high dimensions efficiently.

In order to explain the proposed algorithm, first we need to explain two notions in computational geometry which are the hyperplane distance (Weisstein, 2014a, Weisstein, 2014b) and the convex hull distance.

3.1 Hyperplane Distance

Suppose $V = [v_1, v_2, \dots, v_n]^T$ is a point, F is an *n*-vertex facet, and H is the corresponding hyper-plane of facet F in a *n*-dimensional Euclidean space. Also assume $a_1x_1 + a_2x_2 + \cdots + a_nx_n + b = 0$, is the corresponding equation of H where $N = [a_1, a_2, \dots a_n]^T$ and b are the normal vector and offset, respectively.

The distance from V to the hyperplane H is given by (1).

$$ds(V,H) = \frac{a_1v_1 + a_2v_2 + \cdots a_nv_n + \mathrm{b}}{\sqrt{{a_1}^2 + {a_2}^2 + \cdots {a_n}^2}} \tag{1}$$

3.2 Convex Hull Distance

Given a set $P = \{x_i\}_{i=1}^{n} \subset \mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, the Euclidean distance between x and the convex hull of P, denoted by $conv(P)$, can be computed by solving the quadratic optimization problem stated in (2).

$$\begin{aligned} &\underset{a}{min}\, \frac{1}{2} a^T Q a - c^T a \\ &s.t.\; e^T a = 1, a \geq 0 \end{aligned} \tag{2}$$

Where $e = [1,1,\cdots,1]^T, Q = X^T X$ and $c = X^T x$, with $X = [x_1, x_2, \dots, x_n]$.

Suppose that the optimal solution of (2) is a^*; then the distance of point x to $conv(P)$ is given by:

$$d_c\big(x, conv(P)\big) = \sqrt{x^T x - 2c^T a^* + {a^*}^T Q a^*} \tag{3}$$

3.3 The proposed Algorithm

The proposed algorithm consists of five main steps:

Step 1: Scaling each dimension to the range [-1, 1].

Step 2: Identifying the maximum and minimum samples with respect to each dimension. These samples are considered as vertices of the initial convex hull.

Step 3: Generating a population of k facets based on current vertices of convex hull.

Step 4: Identifying the furthest points to each facet in the current population as new vertices of convex hull, if they have not been detected before.