AN EFFICIENT ALGORITHM FOR DECIDING THE CONVEX SEPARABILITY OF POINT SETS

Gábor TAKÁCS
Advisor: Béla PATAKI

I. Introduction

One of the main goals of *machine learning* is to provide tools for building *intelligent systems*, that gain knowledge, make predictions or make decisions based on a set of data. The heart of such a system is often a *classifier* (e.g. in spam filtering, computer-aided breast cancer recognition, postal code recognition, etc.). The classifier has to guess the value of a high-level discrete attribute (e.g. healthy or sick, spam or nonspam) from the values of some observable, low-level features (e.g. pixel intensities of an X-ray image, words of a text file).

If all the d low-level features are represented by real numbers, then the observations can be treated as points in the d-dimensional Euclidean space. Observations belonging to the same class form a set of points. An important step of building a classification system for a real-life problem is exploring the data. A typical question is whether the classes are *linearly separable* or not. We will see that it can be decided in polynomial time, and there are good practical algorithms performing it.

Another interesting question is whether the classes are *convexly separable* from each other or not. In theory it can be decided in polynomial time too, but conventional methods fail for large tasks in practice due to running time problems. In this paper we will propose an efficient algorithm for deciding the convex separability of point sets. In contrast to the other algorithms available, our method is scalable even for very large problems. We will compare the running time of our algorithm with conventional, linear programming based methods.

II. Notation, Definitions

Let $\mathcal{S}$ be a set of points in the d-dimensional Euclidean space. The *convex hull* of $\mathcal{S}$ denoted by conv($\mathcal{S}$) is the minimal convex set containing $\mathcal{S}$. If $\mathcal{S}$ is finite, then conv($\mathcal{S}$) is a convex polyhedron. A convex polyhedron in d dimensions can be given either by its vertices or its $(d-1)$-dimensional facets. The former is called *vertex representation*, the latter is called *halfspace representation*. The halfspace representation typically cannot be computed in high dimensions, because the number of $(d-1)$-dimensional facets grows exponentially with d.

Let $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m\}$ be two sets of points in $\mathbb{R}^d$. The convex hull of $\mathcal{P}$ and $\mathcal{Q}$ is denoted by conv($\mathcal{P}$) and conv($\mathcal{Q}$).

Definition 1. $\mathcal{P}$ and $\mathcal{Q}$ are called *linearly separable* if $\mathrm{conv}(\mathcal{P}) \cap \mathrm{conv}(\mathcal{Q}) = \emptyset$. Or equivalently: $\exists \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R} : \mathbf{w}^T\mathbf{p}_i + b < 0$ for $i = 1, 2, \ldots, n$ and $\mathbf{w}^T\mathbf{q}_j + b > 0$ for $j = 1, 2, \ldots, m$.

Definition 2. $\mathcal{P}$ and $\mathcal{Q}$ are called *convexly separable* if $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q} = \emptyset$ or $\mathcal{P} \cap \mathrm{conv}(\mathcal{Q}) = \emptyset$. If $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q}$ and $\mathcal{P} \cap \mathrm{conv}(\mathcal{Q})$ are both empty, then $\mathcal{P}$ and $\mathcal{Q}$ are *mutually convexly separable*. If $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q}$ is empty, but $\mathcal{P} \cap \mathrm{conv}(\mathcal{Q})$ is not, then $\mathcal{P}$ is called the *inner set* and $\mathcal{Q}$ the *outer set*.

Note that linear separability implies mutual convex separability, but the reverse is not true.

III. Conventional Methods

A. Linear Separability

Linear separability can be formulated as a *linear programming* (LP) problem as the following:[1]

$$\begin{array}{rcl} \text{variables} & : & \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \\ \text{minimize} & : & 1 \\ \text{subject to} & : & \mathbf{w}^T\mathbf{p}_i + b > 0, \quad i = 1, 2, \ldots, n, \\ & & \mathbf{w}^T\mathbf{q}_j + b < 0, \quad j = 1, 2, \ldots, m. \end{array} \tag{1}$$

$\mathcal{P}$ and $\mathcal{Q}$ are linearly separable if and only if the constraints can be satisfied. Since there exist polynomial algorithms for solving LP ([1, 2]), linear separability can be decided in polynomial time. Problem (1) has no objective function and such a problem is often called a *linear feasibility problem*. Although it might look a simpler problem to solve, it is polynomially equivalent to the general LP to solve it. Unfortunately, this formulation is often unsuitable in practice, because general LP solvers (e.g. MATLAB's) usually fail on large tasks (say $d > 100$ and $n = m > 1000$) due to running time problems.

Fortunately, there is another polynomial formulation, that can handle much larger problem sizes in practice. Linear separability can also be decided with a linear *support vector machine* (SVM) [3]:

$$\begin{array}{rcl} \text{variables} & : & \mathbf{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \\ \text{minimize} & : & \mathbf{w}^T\mathbf{w} \\ \text{subject to} & : & \mathbf{w}^T\mathbf{p}_i + b \geq 1, \quad i = 1, 2, \ldots, n, \\ & & \mathbf{w}^T\mathbf{q}_j + b \leq 1, \quad j = 1, 2, \ldots, m. \end{array} \tag{2}$$

$\mathcal{P}$ and $\mathcal{Q}$ are linearly separable if and only if Problem (2) has a solution. This quadratic programming (QP) problem seems harder than the previous LP one, but for this special type of constrained optimization there exist very efficient solvers (e.g. libsvm [4], svm-light [5]).

B. Convex Separability

$\mathcal{P}$ and $\mathcal{Q}$ are convexly separable if $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q} = \emptyset$ or $\mathcal{P} \cap \mathrm{conv}(\mathcal{Q}) = \emptyset$. Because the two tests are symmetric, assume that we want to decide whether the intersection of $\mathrm{conv}(\mathcal{P})$ and $\mathcal{Q}$ is empty or not. At first we give a simple but exponential method that works well only for small problems:

1. Compute a halfspace representation of $\mathrm{conv}(\mathcal{P})$.
   This yields $\mathbf{w}_k$ -s and b_k -s $(k = 1, 2, \ldots, r)$ such that:
   $\mathrm{conv}(\mathcal{P}) = \{\mathbf{x} \in \mathbb{R}^d : \mathbf{w}_1^T\mathbf{x} \geq b_1,\ \mathbf{w}_2^T\mathbf{x} \geq b_2,\ \ldots,\ \mathbf{w}_r^T\mathbf{x} \geq b_r\}$. (3)
2. $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q} = \emptyset$ if and only if $\max\limits_{j=1,\ldots,m} \{\min\limits_{k=1,\ldots,r} \{\mathbf{w}_k^T\mathbf{q}_j - b_k\}\} < 0$.

The second method is based on LP:

$$\begin{array}{rcl} \text{variables} & : & \mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m \in \mathbb{R}^d,\ b_1, b_2, \ldots, b_m \in \mathbb{R} \\ \text{minimize} & : & 1 \\ \text{subject to} & : & \mathbf{w}_j^T\mathbf{p}_i + b_j > 0, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m, \\ & & \mathbf{w}_j^T\mathbf{q}_j + b_j < 0, \quad j = 1, 2, \ldots, m. \end{array} \tag{4}$$

[1] Strictly speaking this is not an LP, because of the $<$ and $>$ signs. In practice $\leq$ and $\geq$ are used instead, and zeros are replaced to small nonzero numbers.

The necessary and sufficient condition of $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q} = \emptyset$ is the existence of a solution. This method is polynomial in time, but fails for large problems in practice.

The third method is an alternative LP formulation. The LP problem for any fixed $j \in \{1, 2, \ldots, m\}$ is the following:

$$\begin{array}{rcl} \text{variables} & : & \alpha_{j1}, \alpha_{j2}, \ldots, \alpha_{jn} \in \mathbb{R} \\ \text{minimize} & : & 1 \\ \text{subject to} & : & \mathbf{q}_j = \alpha_{j1}\mathbf{p}_1 + \alpha_{j2}\mathbf{p}_2 + \cdots + \alpha_{jn}\mathbf{p}_n, \\ & & \alpha_{j1}, \alpha_{j2}, \ldots, \alpha_{jn} \geq 0, \quad \alpha_{j1} + \alpha_{j2} + \cdots + \alpha_{jn} = 1. \end{array} \tag{5}$$

This formulation exploits the fact that using a vertex representation $\mathrm{conv}(\mathcal{P})$ can be written as $\{\mathbf{x} \in \mathbb{R}^d : \mathbf{x} = \sum_{k=0}^{n} \beta_k \mathbf{p}_k,\ \beta_1, \beta_2, \ldots, \beta_n \geq 0,\ \sum_{k=0}^{n} \beta_k = 1\}$. $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q} = \emptyset$ if and only if Problem (5) is infeasible for any $j \in \{1, 2, \ldots, m\}$.

IV. The Proposed Method

At first we introduce a fast algorithm that performs approximate convex separation:

1. Initialize signed distances with $s_1 = s_2 = \cdots = s_m = \infty$.
2. Compute the centroid of $\mathcal{P}$ as $\bar{\mathbf{p}} = \frac{1}{n}(\mathbf{p}_1 + \mathbf{p}_2 + \cdots + \mathbf{p}_n)$.
3. Choose $\mathbf{q}_k$ from $\mathcal{Q}$ such that $k = \arg\max_{j=1,\ldots,m}\{s_j\}$.
4. If $s_k \leq 0$, then stop.
5. Compute $\mathbf{w} = (\bar{\mathbf{p}} - \mathbf{q}_k)/\|\bar{\mathbf{p}} - \mathbf{q}_k\|$ and $b = \min_{i=1\ldots n}\{\mathbf{w}^T\mathbf{p}_i\}$.
6. $s_k \leftarrow 0$.
7. For all $s_j > 0 : s_j \leftarrow \min\{s_j,\ \mathbf{w}^T\mathbf{q}_j - b\}$.
8. Go to step 3.

(6)

This algorithm translates at most m hyperplanes to their "optimal" positions. (The number of q_j-s on negative side of the hyperplane is maximized while all p_i-s have to be on the positive side.) The algorithm can be called the *centroid method*, because the hyperplanes are defined by connecting the centroid of $\mathcal{P}$ with the elements of $\mathcal{Q}$. This extremely simple and fast method is often able to separate most of the elements of $\mathcal{Q}$ from $\mathrm{conv}(\mathcal{P})$. The disadvantage of the algorithm is that it does not guarantee to find a convex separation for convexly separable problems.

At second we straightforwardly extend the SVM method to the case of convex separability:

$$\begin{array}{rcl} \text{variables} & : & \mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_m \in \mathbb{R}^d,\ b_1, b_2, \ldots b_m \in \mathbb{R} \\ \text{minimize} & : & \mathbf{w}_1^T\mathbf{w}_1,\ \mathbf{w}_2^T\mathbf{w}_2,\ \ldots,\ \mathbf{w}_m^T\mathbf{w}_m \\ \text{subject to} & : & \mathbf{w}_j^T\mathbf{p}_i + b_j \geq 1, \quad i = 1, 2, \ldots n, \quad j = 1, 2, \ldots m, \\ & & \mathbf{w}_j^T\mathbf{q}_j + b_j \leq -1, \quad j = 1, 2, \ldots m. \end{array} \tag{7}$$

$\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q} = \emptyset$ if and only if the above m constrained optimization problems all have a solution. The disadvantage of this method is enormous running time for large m. (However it is faster than the conventional LP-based method.)

A very efficient algorithm can be obtained by combining the previous two methods. At first the size of $\mathcal{Q}$ is reduced by applying the centroid method. Denote the set of unseparated $\mathbf{q}_j$-s by $\mathcal{Q}_U$. At second $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q}_U = \emptyset$ is checked by running the SVM method.

Table 1: Running times. (N/A: no answer in reasonable time.)

	d	n	m	LP1	LP2	SVM	Centroid+SVM
Votes	16	267	168	33.0 s	15.9 s	0.25 s	0.15 s
Wisconsin	9	444	239	63.9 s	23.4 s	0.40 s	0.39 s
MNIST01	196	6903	7877	N/A	N/A	25.3 s	0.86 s
MNIST02	196	6903	6990	N/A	N/A	325.8 s	11.4 s
MNIST79	196	7693	6958	N/A	N/A	919.0 s	145.8 s
MNIST49	196	6824	6958	N/A	N/A	1235.1 s	268.2 s

V. Comparison

We compared the running time of our method with conventional LP-based methods in 6 real-world problems. The first two test problems originate from the UCI machine learning repository [6]. The *Voting Records* dataset includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes. The class attribute is whether the voter is democrat or republican. The *Wisconsin Breast Cancer* dataset contains measurements on 239 malignant tumors and 444 benign findings in women's breast. Both datasets are convexly separable, but not mutually.

The other test problems were extracted from the the *MNIST* handwritten digit recognition database [7]. There is a linearly separable (digit 0 vs. digit 1), a mutually convexly separable (0 vs. 2), a convexly separable (7 vs. 9) and a convexly nonseparable (4 vs. 9) among them.

The algorithms had to return the number of elements in $\mathrm{conv}(\mathcal{P}) \cap \mathcal{Q}$ and $\mathcal{P} \cap \mathrm{conv}(\mathcal{Q})$, because this information is more useful in practice than a plain yes-no answer. Fortunately, all algorithms can easily be modified to return these 2 numbers instead of a boolean value. The algorithms in the comparison were the following:

- LP1: The LP-based formulation given in (4), implemented in MATLAB.
- LP2: The alternative LP-based formulation given in (5), implemented in MATLAB.
- SVM: The SVM-based formulation given in (7), implemented in Java using libsvm [4].
- Centroid+SVM: The proposed algorithm, implemented in Java using libsvm.

The results can be seen in Table 1.

VI. Conclusion

Convex separability is an interesting question in the data exploration phase of building classification systems for real-life problems. In this paper we proposed an efficient algorithm for deciding the convex separability of two point sets in $\mathbb{R}^d$. Our algorithm was more than 50 times faster than conventional LP-based methods in all test problems.

References

[1] L. Khachiyan, "Polynomial algorithm for linear programming," *Dokl. Akad. Nauk SSSR*, 224:1093–1096, 1979.

[2] N. Karkamar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, 4:373–395, 1984.

[3] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[4] C. Chang and C. Lin, *LIBSVM: a library for support vector machines*, National Taiwan University, 2001, Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[5] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. MIT Press, 1999.

[6] D. Newman, S. Hettich, C. Blake, and C. Merz, *UCI repository of machine learning databases*, University of California, Irvine, CA, 1998, Available at http://www.ics.uci.edu/∼mlearn/MLRepository.html.

[7] Y. LeCun, *The MNIST database of handwritten digits*, 2003, Available at http://yann.lecun.com/exdb/mnist.