An Algorithm to Compute the Distance from a Point to a Simplex

Oleg Golubitsky, Vadim Mazalov and Stephen M. Watt

Introduction

We present an efficient algorithm to compute the distance from a point to a simplex in n-dimensional Euclidean space. The method is based on recursive projection onto linear subspaces containing lower-dimensional subsimplexes, proceeding until the projection is in the interior. This method has application to classification problems based on distances to convex hulls of points, which frequently appear in pattern recognition.

The Problem

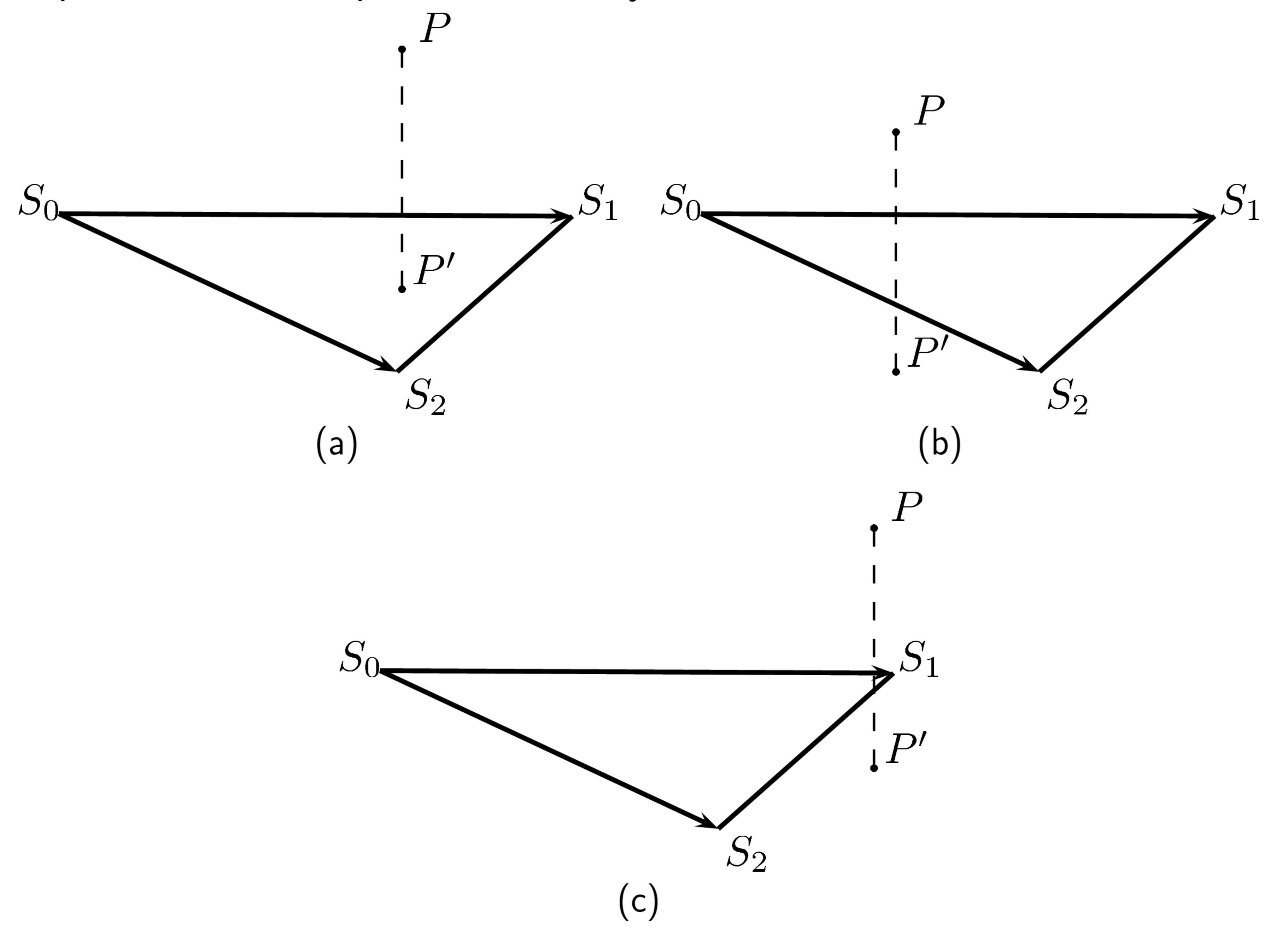
The task of computing the distance from a point to a convex hull of k nearest neighbours occurs in various applications of machine learning. Since any generic set of points of size not exceeding the dimension of the vector space plus one forms a simplex, we view the distance to the convex hull of k points as the distance to a simplex. There is some research of solving more general or more specific problems [1,2,3,4], but this particular task deserves further attention. We show how the distance from a point to a general simplex may be solved simply.

Main Idea

The main idea of our method is the following: If the point is "above" a face (i.e. it falls within the prism generated by translating the face orthogonally), then the closest point is the projection onto the interior of the face. Otherwise, the closest point will lie on one of the $(n-1)$ dimensional edge facets of the closest face, and the problem can be solved recursively.

Algorithm

The method we present is similar to that of Michelot [3], except that it computes the distance to a general simplex. While this could be accomplished by finding a linear transformation mapping the general simplex to the canonical one, and conjugating the Michelot's method, it is simpler to perform the computation directly.



Possible projection scenarios

The projection from the point to the smallest linear subspace that contains the simplex is expressed as a linear combination of the generating vectors of the simplex. The vectors with corresponding positive coefficients are used as the input for the next recursive call. The algorithm stops when all of the coefficients of a projection are non-negative or when the simplex contains only one vertex.

Let $S_i \in \mathbb{R}^n, i = 0..d, d \geq 0, d \leq n$ be points of a simplex and $P \in \mathbb{R}^n$ is the point from which the distance should be computed, where $\mathbb{R}^n$ is the n-dimensional Euclidean space. We assume that the points of the simplex are in generic position, i.e. the vectors $\mathbf{S_1} - \mathbf{S_0}, \mathbf{S_2} - \mathbf{S_0}, ..., \mathbf{S_d} - \mathbf{S_0}$ are linearly independent. Otherwise, one may perform simplicial decomposition of their convex hull. With the assumption, see the Algorithm for a formal description.

```
Algorithm DistanceToSimplex($P, \{S_0, ..., S_d\}$).
Input: A point $P$ and a simplex with vertices $\{S_0, ..., S_d\}$.
Output: Distance from $P$ to the simplex.
  if $d = 0$ then
    return Euclidean distance between $P$ and $S_0$.
  end if
  Translate so that $S_0$ is the origin.
  Find projection $P'$ of $P$ to the linear subspace with the set of basis vectors
  $\mathbf{S} = \mathbf{S_1}, ..., \mathbf{S_d}$. The projection can be computed as a solution of the system
      $$\sum_{i=1}^{d} \alpha_i \langle \mathbf{S_i}, \mathbf{S_j} \rangle = \langle \mathbf{P}, \mathbf{S_j} \rangle, j = 1, 2, ..., d$$
  and expressed as $\mathbf{P}' = \sum_{i=1}^{d} \alpha_i \mathbf{S_i}$.
  if $\sum_{i=1}^{d} \alpha_i \leq 1$ and $\alpha_i \geq 0, \forall i = 1..d$ then
    {The projection is inside the simplex, see Figure (a)}
    return Euclidean distance between $P$ and $P'$.
  else if $\exists i$ such that $\alpha_i < 0$ then
    {See Figure (b)}
    $S' \leftarrow S_0 \cup \{S_i | \alpha_i > 0\}$.
    return DistanceToSimplex($P, S'$)
  else
    {$\sum_{i=1}^{d} \alpha_i > 1$ and $\alpha_i \geq 0, \forall i = 1..d$, see Figure (c)}
    return DistanceToSimplex($P, S \setminus S_0$)
  end if
```

The complexity of the algorithm is $O(d^4)$, where d is the number of vertices. In practice, however, the algorithm performs much faster, since on each recursive call the dimension typically drops by more than one.

Recursive call	S	α
1	$\{S_0(0,0), S_1(3,0), S_2(2,-1)\}$	$\alpha_1 = 2, \alpha_2 = -1$
2	$\{S_0(0,0), S_1(3,0)\}$	$\alpha_1 = \frac{4}{3}$
3	$\{S_0(3,0)\}$	

An example of projecting a point $P(4,1)$ to the simplex $S = \{S_0(0,0), S_1(3,0), S_2(2,-1)\}$

References

1. V. Pascal, Y. Bengio, K-Local Hyperplane and Convex Distance Nearest Neighbor Algorithms. Advances in Neural Information Processing Systems 2002, V. 2, MIT Press, pp. 985-992.
2. P. Wolfe, Finding the Nearest Point in A Polytope, Mathematical Programming, 1976, V. 11, N. 1, Springer Berlin / Heidelberg, pp. 128-149.
3. C. Michelot, A Finite Algorithm for Finding the Projection of a Point onto the Canonical Simplex of $\mathbb{R}^n$, J. Optimization Theory and Applications 1986, V. 50, N. 1, pp. 195-200.
4. D. Wilhelmsen, A Nearest Point Algorithm for Convex Polyhedral Cones and Applications to Positive Linear Approximation, Mathematics of Computation, 1976, V. 30, N. 133, pp. 48-57.