

Lecture 1: Convex Hulls I

Lecturer: Suresh Venkatasubramanian

Scribe:

1.1 Convex Sets and Hulls

Definition 1.1.1 (Convex set) A set $A \in \mathbb{R}^d$ is convex if for any two points $p, q \in A$, and any $0 \leq \lambda \leq 1$, $\lambda p + (1 - \lambda)q \in A$.

In other words, for any two points in A , a straight line drawn between them must lie wholly in A . The two shapes in Figure 1.1 illustrate this idea.

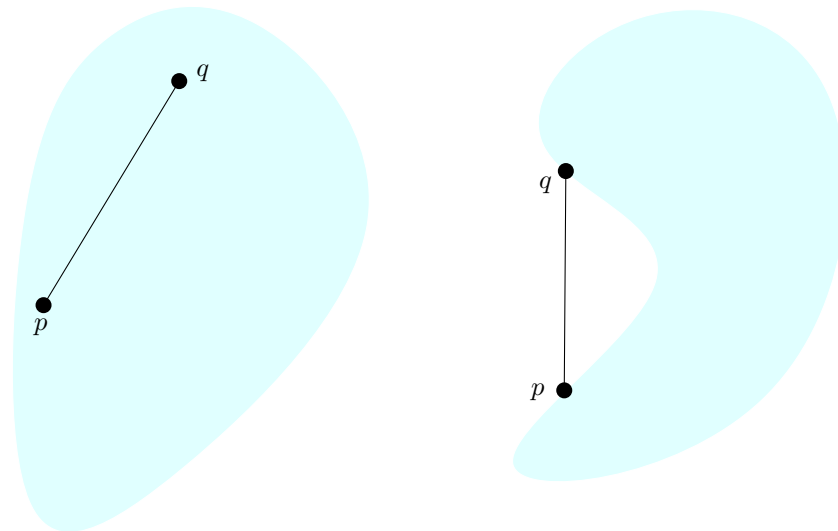


Figure 1.1: Examples of convex and non-convex sets

This notion of convexity can be used to define the convexity of a function.

Definition 1.1.2 (Convex function) A real-valued function f defined on an interval (or on some convex region of space) is convex if the set

$$\{(x, y) \mid y \geq f(x)\}$$

is convex.

Now we can define the convex hull of a set of points.

Definition 1.1.3 (Convex Hull) *The convex hull $\mathcal{CH}(P)$ of a set of n points P is the smallest convex set containing P . Equivalently, $\mathcal{CH}(P)$ is the intersection of all convex sets containing P .*

The intersection of convex sets is convex, so the convex hull is a convex set. It is also a *polytope*, a bounded intersection of halfspaces, which in two dimensions means that it is a polygon. Another useful characterization of the convex hull is as a *convex closure*: take all possible convex combinations of points in P , and the result is the convex hull.

$$\mathcal{CH}(P) = \left\{ \sum_{i=1}^n \lambda_i p_i \mid \lambda_i \geq 0, \sum_i \lambda_i = 1, p_i \in P \right\}$$

In two dimensions, the convex hull has size at most n . In higher dimensions, its size is governed by the Upper Bound Theorem, which we shall see in the next lecture.

1.2 Computing A Planar Convex Hull

Representation Since the convex hull is a polygon, a natural representation of the hull is as an ordered list of the vertices, in clockwise or counter-clockwise order. A result by Yao[Yao81] shows that even if we merely required the set of vertices in no particular order, the problem remains as hard (requiring $\Omega(n \log n)$ time).

In all the algorithms that follow, we will assume that the output is an ordered sequence of points on the hull.

1.2.1 A Naive Algorithm

The convex hull of a planar point set P consists of line segments connecting certain points in P . Under what condition is the line segment $\overline{pp'}$ connecting p, p' an element of the convex hull boundary?

Fact 1 $\overline{pp'}$ is on the convex hull iff all remaining points of P lie on one of its sides.

To check which side of a line a point lies on, we substitute the coordinates of the point into the equation of the line, and record the sign of the answer. For example, given the line $x + 2y - 2 = 0$ that passes through the points $(2, 0)$ and $(0, 1)$, consider what happens when we test the points $(0, 0)$ and $(1, 1)$. These points lie on opposite sides of the line segment, and indeed when we substitute, we get -2 and $+1$, which have different sign.

For each pair of points in P , we can check whether the line segment connecting them lies on the convex hull. This check takes $n - 1$ steps for each pair, and there are $\binom{n}{2}$ pairs of points. Therefore, the overall running time of this naive algorithm is $O(n^3)$.

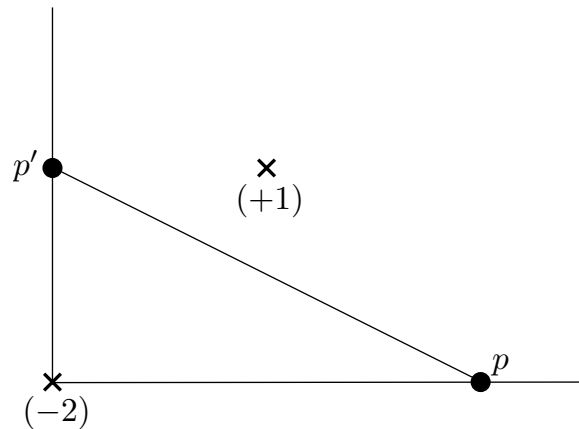


Figure 1.2: Checking which side of a line a point lies on

1.2.2 The Jarvis March

It's not too difficult to see that this is very wasteful. For example, suppose we've already found a few line segments that form part of the convex hull, and that they form a connected section, shown in Figure 1.3

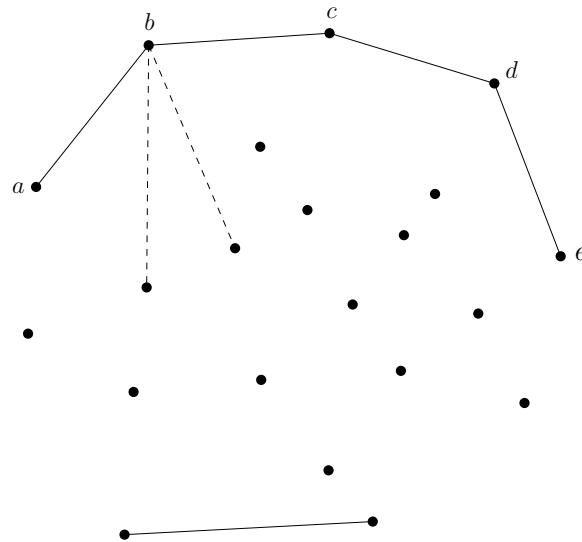


Figure 1.3: Pruning unnecessary line segments in the naive algorithm

It is now clear that any line segment terminating in the point b can never be part of the convex hull. In fact, the only line segments that need consideration are those that terminate in the endpoints of the current hull; for example, in vertices a and e .

This observation implies that we can merely *grow* a partial hull at its boundaries, which will

yield a new convex hull vertex after $O(n)$ operations. Thus, the overall running time reduces to $O(n^2)$.

But one important step remains. How do we start the process? We need to find at least one vertex that is guaranteed to lie on the convex hull. You can convince yourself that any extreme point (leftmost, rightmost, topmost, bottommost) must lie on the convex hull. In fact, if a point is extreme *in any direction*, it must lie on the convex hull (**Why?**).

We can easily compute an extreme point p_0 in $O(n)$ time, and then we can start the growing process. We find a line segment $\overline{p_0 p_i}$ that passes the all-points-on-one-side test, add p_i to the ordered list, and continue from p_i .

This algorithm is known as the *Jarvis March*, after the person who discovered it in 1973[[Jar73](#)]. It is also often called the *gift-wrapping* algorithm, since the algorithm proceeds by “wrapping” a line around the convex hull.

Output-Sensitivity We can actually perform a slightly more nuanced analysis of the Jarvis March, if we introduce the parameter h , the actual size of the convex hull (since it’s a polygon, this can be either the number of vertices or the number of edges). In each linear-time step, one new convex hull vertex is added. Therefore, the running time can be rewritten as $O(nh)$. Such a running time, expressed both in terms of the input size (n) and the output size (h), is called *output-sensitive*; the algorithm is said to be *output-sensitive*.

Note that in the worst-case, $h = n$, so this bound reduces to $O(n^2)$. But for instances where the convex hull might be a triangle or a quadrilateral (for example), this algorithm is quite efficient.

1.2.3 The Graham Scan (as modified by Andrews)

It turns out that we can do better than $O(n^2)$, using an algorithm first developed by Graham, and then modified by Andrews. It’s called the Graham scan, and is described in detail in the textbook[[dBvKOS00](#), Chapter 1]. The running time of this algorithm is $O(n \log n)$, which is optimal in the worst-case. Note that if the input point set has a small convex hull ($h \leq \log n$), then the Jarvis march is superior to the Graham scan.

1.2.4 Robustness

Both the Graham scan and the Jarvis march make use of a sign-testing step. In the Jarvis march, we check whether a point is on a particular side of a line by evaluating the sign of a *linear* function of the point’s coordinates. In the Graham scan, we need to determine the signed area of a triangle, which involves a *quadratic* sign test.

These tests are very sensitive, since whether the expression is greater than zero or not changes the behaviour of the algorithm. In theoretical descriptions of geometric algorithms, we tend to ignore this issue, pretending that all inputs come with infinite precision, and that we can evaluate such expressions exactly.

In practice, this is never the case; most often, we use 32-bit floating point numbers to represent point coordinates, and sometimes we use 64-bit `double` representations. The use of limited precision means that often such sign tests will yield incorrect results. For example, if a point is very close to a line, an imprecise sign test may not be able to detect which side it is on correctly.

Software packages like CGAL[[cga](http://www.cgal.org)] incorporate methods to deal with the limited precision of computer representations of numbers when executing geometric algorithms.

References

- [[cga](http://www.cgal.org)] Cgal. <http://www.cgal.org>.
- [dBvKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2000.
- [Jar73] R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Inform. Process. Lett.*, 2:18–21, 1973.
- [Yao81] Andrew Chi-Chih Yao. A lower bound to finding convex hulls. *J. ACM*, 28(4):780–787, 1981.