Convex Sets, Convex Hull

Coman Florin-Alexandru

April 2, 2015

Table of contents

Convex Sets

Convex Sets

Intuitively, if we think of $\mathbb{R}^2$ or $\mathbb{R}^3$, a convex set of vectors is a set that contains all the points of any line segment joining two points of the set.

Examples

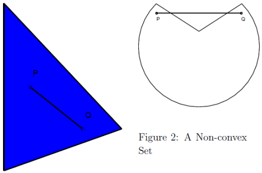


Figure 1: A Convex Set

Figure 2: A Non-convex Set

Convex Sets

Convex Combinations - Definition

Let $u, v \in V$. Then the set of all **convex combinations** of u and v is the set of points

$$\{\omega_\lambda \in V : \omega_\lambda = (1-\lambda)u + \lambda v, 0 \leq \lambda \leq 1\} \qquad (1.1)$$

For example, in $\mathbb{R}^2$, this set is exactly the line segment joining the two points u and v.

Convex Sets

Definition

Let $K \subset V$. Then the set K is said to be **convex** provided that given two points $u, v \in K$ the set (1.1) is a subset of K.

Example

An interval $[a, b] \subset \mathbb{R}$ is a convex set.
To see this, let $c, d \in [a, b]$ and assume, without loss of generality, that $c < d$. Let $\lambda \in (0, 1)$. Then,

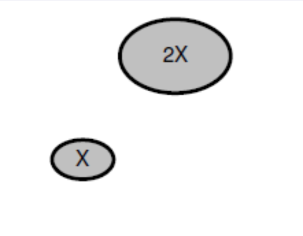
$$
\begin{aligned}
a \leq c &= (1-\lambda)c + \lambda c < (1-\lambda)c + \lambda d \\
&\leq (1-\lambda)d + \lambda d = d \\
&\leq b.
\end{aligned}
$$

Convex Sets

Proposition 1.1

If X is a convex set and $\beta \in \mathbb{R}$, the set $\beta X = \{y : y = \beta x, x \in X\}$ is convex.

Proof

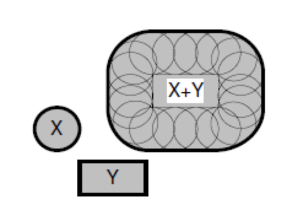


Convex Sets

Proposition 1.2

If X and Y are convex sets, then the set
$X + Y = \{z : z = x + y, x \in X, y \in Y\}$ is convex.

Proof

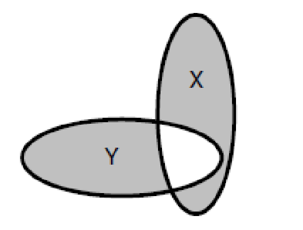


Convex Sets

Proposition 1.3

The intersection of any collection of convex sets is convex.

Proof



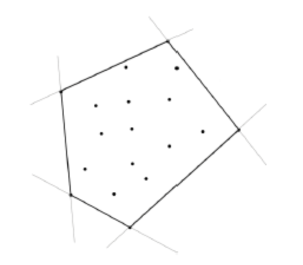
Convex Hull

Definition

The **convex hull** of a set C is the intersection of all convex sets which contain the set C.

Convex Hull

Example



Representation of a convex set as the convex hull of a set of points.

Convex Hulls - Algorithms

The Jarvis March Algorithm

```
jarvis(S)
    pointOnHull = leftmost point in S
    i = 0
    repeat
        P[i] = pointOnHull
        endpoint = S[0] /* initial endpoint for a candidate edge on the hull */
        for j from 1 to |S|
            if (endpoint == pointOnHull) or (S[j] is on left of line from P[i] to endpoint)
                endpoint = S[j] /* found greater left turn, update endpoint */
        i = i+1
        pointOnHull = endpoint
    until endpoint == P[0] /* wrapped around to first hull point*/
```

Convex Hulls - Algorithms

The Andrew's Algorithm

```
Input:  a list P of points in the plane.
Sort the points of P by x-coordinate (in case of a tie, sort by y-coordinate).
Initialize U and L as empty lists.
The lists will hold the vertices of upper and lower hulls respectively.
```

```
for i = 1, 2, ..., n:
    while L contains at least two points and the sequence of last two points of L and the
            point P[i] doesn't make a counter-clockwise turn:
        remove the last point from L
    append P[i] to L
for i = n, n-1, ..., 1:
    while U contains at least two points and the sequence of last two points of U and the
            point P[i] doesn't make a counter-clockwise turn:
        remove the last point from U
    append P[i] to U
```

Remove the last point of each list (it's the same as the first point of the other list).
Concatenate L and U to obtain the convex hull of P.
Points in the result will be listed in counter-clockwise order.

Convex Hulls - Algorithms

FindHull(P, m)

Partition P into n/m groups P_i
for i = 1 ... n/m:
        $\mathcal{H}_i$ = ConvexHull(P_i)
Run Jarvis march on $\{\mathcal{H}_i\}$ for m steps
if we get a complete hull then
        return success
else
        return fail

Chan's Algorithm

i = 0
while FindHull($P, 2^{2^i}$) fails
        i = i + 1

Convex Hulls - Algorithms

Quickhull Algorithm

1. Find the points with minimum and maximum x coordinates, those are bound to be part of the convex hull.
2. Use the line formed by the two points to divide the set in two subsets of points, which will be processed recursively.
3. Determine the point, on one side of the line, with the maximum distance from the line. The two points found before along with this one form a triangle.
4. The points lying inside of that triangle cannot be part of the convex hull and can therefore be ignored in the next steps.
5. Repeat the previous two steps on the two lines formed by the triangle (not the initial line).
6. Keep on doing so on until no more points are left, the recursion has come to an end and the points selected constitute the convex hull.

Convex Hulls - Algorithms

Graham scan algorithm I

```
/* Three points are a counter-clockwise turn if ccw > 0, clockwise if ccw < 0, and collinear if
ccw = 0 because ccw is a determinant that gives twice the signed area of the triangle formed by
p1, p2 and p3.*/
function ccw(p1, p2, p3):
        return (p2.x - p1.x)*(p3.y - p1.y) - (p2.y - p1.y)*(p3.x - p1.x)
```

Convex Hulls - Algorithms

Graham scan algorithm II

```
N = number of points
points[N+1] = the array of points
swap points[1] with the point with the lowest y-coordinate sort points by polar angle with points[1]
/* We want points[0] to be a sentinel point that will stop the loop. */
points[0] = points[N]
/* M will denote the number of points on the convex hull. */
M = 1
for i = 2 to N:
        /* Find next valid point on convex hull. */
        while ccw(points[M-1], points[M], points[i]) ¡= 0:
            if M ¿ 1:
                M -= 1
            /* All points are collinear */
            else if i == N:
                break
            else
                i += 1
        /* Update M and swap points[i] to the correct place. */
        M += 1
        swap points[M] with points[i]
```

Bibliography I

[1] Levent Kandiller, *Principles of Mathematics in Operations Research*, The International Series in Operations Research and Management Science, Vol. 97, Springer, 2007.

[2] Geoffrey J. Gordon, *Lecture 3. Convex sets*, Optimization Course, CMU, Fall 2012.

[3] Anatoli Juditsky, *Lecture 1. Convex sets*, Convex Optimization, Theory and Algorithmes Course, Laboratoire Jean Kuntzmann

[4] Arne Hallam, *Convex Sets*, Convexity and Optimization Course, Iowa State University