Basic Concepts in Convexity and Computation

Jesús A. De Loera, UC Davis

July 15, 2009
Combinatorial Convexity
Everything we do takes place inside $\mathbb{R}^d$. We have the traditional Euclidean inner-product, norm of vectors, and distance between two points $x, y$ defined by $\sqrt{(x_1 - y_1)^2 + \ldots + (x_2 - y_2)^2}$. The set of all points $[x, y] := \{\alpha x + (1 - \alpha)y : 0 \leq \alpha \leq 1\}$ is called the line segment between $x$ and $y$. The points $x$ and $y$ are the endpoints of the interval. A subset $S$ of $\mathbb{R}^n$ is called convex if for any two distinct points $x_1, x_2$ in $S$ the line segment joining $x_1, x_2$, lies completely in $S$. 
A linear functional $f : \mathbb{R}^d \to \mathbb{R}$ is given by a vector $c \in \mathbb{R}^d$, $c \neq 0$.

For a number $\alpha \in \mathbb{R}$ we say that $H_\alpha = \{x \in \mathbb{R}^d : f(x) = \alpha\}$ is an affine hyperplane or hyperplane for short.

The intersection of finitely many hyperplanes is an affine space. Affine spaces are convex, but always contain lines. The affine hull of a set $A$ is the smallest affine space containing $A$.

Note that a hyperplane divides $\mathbb{R}^d$ into two halfspaces $H_\alpha^+ = \{x \in \mathbb{R}^d : f(x) \geq \alpha\}$ and $H_\alpha^- = \{x \in \mathbb{R}^d : f(x) \leq \alpha\}$. Halfspaces are convex sets.
The intersection of finitely many half-spaces is a **polyhedron**.

Similarly: A polyhedron is then the set of solutions of a system of linear inequalities

\[ P = \{ x \in \mathbb{R}^d : < c_i, x > \leq \beta_i \}, \]

for some non-zero vectors \( c_i \) in \( \mathbb{R}^d \) and some real numbers \( \beta_i \).

The intersection of convex sets is always convex. Let \( A \subset \mathbb{R}^d \), the convex hull of \( A \), denoted by \( \text{conv}(A) \), is the intersection of all the convex sets containing \( A \).

A **polytope** is the convex hull of a finite set of points in \( \mathbb{R}^d \). It is the smallest convex set containing the points.

The image of a convex set under a linear transformation is again a convex set.

A polyhedron (polytope) is always a convex set. A linear image of a polyhedron (polytope) is always convex. How are they related?
Definition: Given finitely many points $A := \{x_1, x_2, \ldots, x_n\}$ we say the linear combination $\sum \gamma_i x_i$ is
- an affine combination if $\sum \gamma_i = 1$.
- a convex combination if it is affine and $\gamma_i \geq 0$ for all $i$.

Lemma: For a set of points $A$ in $\mathbb{R}^d$ we have that $\text{conv}(A)$ equals all finite convex combinations of $A$:

$$\text{conv}(A) = \left\{ \sum_{x_i \in A} \gamma_i x_i : \gamma_i \geq 0 \text{ and } \gamma_1 + \ldots + \gamma_k = 1 \right\}$$

We say a set of points $x_1, \ldots, x_n$ is affinely dependent if there is a linear combination $\sum a_i x_i = 0$ with $\sum a_i = 0$. Otherwise we say they are affinely independent.

Lemma: A set of $d + 2$ or more points in $\mathbb{R}^d$ is affinely dependent.

Lemma: A set $B \in \text{real}^d$ is affinely independent if and only if every point has a unique representation as an affine combination of points in $B$. 
Theorem: (Caratheodory’s theorem): If \( x \in \text{conv}(S) \subset \mathbb{R}^d \), then \( x \) is the convex combination of \( d + 1 \) points.

Theorem: (Radon’s theorem): If a set \( A \) with \( d + 2 \) points in \( \mathbb{R}^d \) then \( A \) can be partitioned into two sets \( X, Y \) such that \( \text{conv}(X) \cap \text{conv}(Y) \neq \emptyset \).

Theorem: (Helly’s theorem): If \( C \) is a collection of closed bounded convex sets in \( \mathbb{R}^d \) such that each \( d + 1 \) sets have nonempty intersection then the intersection of all sets in \( C \) is non-empty.
For a convex set $S$ in $\mathbb{R}^d$. A linear inequality $f(x) \leq \alpha$ is said to be valid on $S$ if every point in $P$ satisfies it.

A set $F \subset S$ is a face of $P$ if and only there exists a linear inequality $f(x) \leq \alpha$ which is valid on $P$ and such that $F = \{x \in P : f(x) = \alpha\}$. Then the hyperplane defined by $f$ is a supporting hyperplane of $F$.

The dimension of an affine set is the largest number of affinely independent points in the set minus one. The dimension of a set in $\mathbb{R}^d$ is the dimension of its affine hull.

A face of dimension 0 is called a vertex. A face of dimension 1 is called an edge, and a face of dimension $\dim(P) - 1$ is called a facet. The empty set is defined to be a face of $P$ of dimension $-1$. Faces that are not the empty set or $P$ itself are called proper.
Lemma: Let $P = \text{conv}(a_1, \ldots, a_n)$ be a polytope and $F \subset P$ a face of $P$. Then $F = \text{conv}(a_i, a_i \in F)$.

Corollary: A Polytope has a finite number of faces, in particular a finite number of vertices and facets.

Lemma Let $P$ be a $d$-polytope and $F \subset P$ be a face. Let $G \subset F$ be a face of $F$. Then $G$ is a face of $P$. Faces form a partially ordered set by containment face poset of a polytope.

Theorem Let $Q = \{x : Ax \leq b\}$ a polyhedron. A non-empty subset $F$ is a face of $P$ if and only if $F$ is the set of solutions of a system of inequalities and equalities obtained from the list $Ax \leq b$ by changing some of the inequalities to equalities.

Corollary: The set of of faces of a polyhedron forms also a poset by containment and it is finite.
**Definition:** Two polytopes are **combinatorially isomorphic** if their face posets are the same.

It follows: two polytopes $P, Q$ are isomorphic if there is a one-to-one correspondence $p_i$ to $q_i$ between the vertices such that $\text{conv}(p_i : i \in I)$ is a face of $P$ if and only if $\text{conv}(q_i : i \in I)$ is a face of $Q$.

**Definition:** The **graph of a polytope** (or polyhedron) is the graph given of 1-dimensional faces (edges) and the vertices (0-dimensional faces).

**Theorem:** *(Balinski’s theorem)* The graphs of $d$-dimensional polytopes are always $d$-connected.
For $A \subset \mathbb{R}^d$ polar of $A$ is

$$A^o = \{x \in \mathbb{R}^d : \langle x, a \rangle \leq 1 \text{ for every } a \in A\}$$

Another way of thinking of the polar is as the intersection of the halfspaces, one for each element $a \in A$, of the form

$$\{x \in \mathbb{R}^d : \langle x, a \rangle \leq 1\}$$

**Example 1:** Take $L$ a line in $\mathbb{R}^2$ passing through the origin, what is $L^0$? the perpendicular line that passes through the origin.

**Example 2:** If the line $L$ does not pass through the origin then, $L^0$ is a clipped line orthogonal to the given line that passes through the origin.
Theorem For any polytope $P$, there is a polytope $P^*$, a dual polytope of $P$, whose face lattice is isomorphic to the reversed poset of the face lattice of $P$.

idea of proof: Translate $P \subset \mathbb{R}^d$ to contain the origin as its interior point. For a non-empty face $F$ of $P$ define

$$\hat{F} = \{ x \in P^o : <x, y> = 1 \text{ for all } y \in F \}$$

and for the empty face define $\hat{\emptyset} = P^0$.

The hat operation applied to faces of a $d$-polytope $P$ satisfies

1. The set $\hat{F}$ is a face of $P^o$
2. $\dim(F) + \dim(\hat{F}) = d - 1$.
3. The hat operation is involutory: $\hat{\hat{F}} = F$.
4. If $F, G \subset P$ are faces and $F \subset G \subset P$, then $\hat{G}, \hat{F}$ are faces of $P^o$ and $\hat{G} \subset \hat{F}$. 
For any $d$-polytope, denote by $f_i(P)$ the number of $i$-faces of $P$. The $f$-vector of $P$ is

$$f(P) = (f_0(P), f_1(P), \ldots, f_{d-1}(P)).$$

**Theorem** (Euler-Poincaré formula) For any $d$-dimensional Polytope $P$, then

$$\sum_{i=-1}^{d} (-1)^i f_i(P) = 0$$

**Theorem** (Upper bound theorem) For any $d$-polytope $P$ with $n$ vertices has no

$$f_i(P) \leq f_i(C(n, d))$$

Where $C(n, d)$ is a special polytope, the cyclic polytope. In particular $f_{d-1}(P) \leq O(n^{\lfloor d/2 \rfloor})$. 
Convexity XII

- **Theorem:** [Weyl-Minkowski] Every polytope is a polyhedron. Every bounded polyhedron is a polytope.
- This allows us to represent all polytopes in two ways inside a computer!! Either as a list of vertices, or as system of inequalities.
Computational Complexity
Computational Complexity

- We need a theory to measure how difficult is to compute!!!
We need a theory to measure how difficult is to compute!!!

The study of the **efficiency** of algorithms and the **difficulty** of problems.
Computational Complexity

- We need a theory to measure how difficult is to compute!!!
- The study of the efficiency of algorithms and the difficulty of problems.
- **Problem:** a generic computational question that can be stated without any data specified.
We need a theory to measure how difficult is to compute!!!
The study of the \textit{efficiency} of algorithms and the \textit{difficulty} of problems.

\textbf{Problem}: a generic computational question that can be stated without any data specified.

\textbf{Examples}:
- \textsc{Clique}: Given a graph \( G = (V, E) \) and an integer \( k \), does \( G \) have a clique of size \( k \)?
Computational Complexity

- We need a theory to measure how difficult is to compute!!!
- The study of the **efficiency** of algorithms and the **difficulty** of problems.
- **Problem:** a generic computational question that can be stated without any data specified.
- **Examples:**
  - **CLIQUE:** Given a graph $G = (V, E)$ and an integer $k$, does $G$ have a clique of size $k$?
- An **instance** of a problem is a particular specification of the data.
We need a theory to measure how difficult it is to compute!!!

The study of the **efficiency** of algorithms and the **difficulty** of problems.

**Problem:** a generic computational question that can be stated without any data specified.

**Examples:**

- **Clique:** Given a graph \( G = (V, E) \) and an integer \( k \), does \( G \) have a clique of size \( k \)?

An **instance** of a problem is a particular specification of the data.

**Examples:**
Computational Complexity

- We need a theory to measure how difficult is to compute!!!
- The study of the efficiency of algorithms and the difficulty of problems.

**Problem:** a generic computational question that can be stated without any data specified.

**Examples:**
- **Clique:** Given a graph \( G = (V, E) \) and an integer \( k \), does \( G \) have a clique of size \( k \)?

An **instance** of a problem is a particular specification of the data.

**Examples:**
We need a theory to measure how difficult is to compute!!!

The study of the efficiency of algorithms and the difficulty of problems.

**Problem:** a generic computational question that can be stated without any data specified.

**Examples:**

- **CLIQUE:** Given a graph $G = (V, E)$ and an integer $k$, does $G$ have a clique of size $k$?

An instance of a problem is a particular specification of the data.

**Examples:**

- Does the Peterson graph have a clique of size 4?
An algorithm is a finite set of instructions for performing basic operations on an input to produce an output.
Algorithms

- An algorithm is a finite set of instructions for performing basic operations on an input to produce an output.
- An algorithm \( A \) solves a problem \( P \) if given a representation of each instance \( I \) as input it supplies as output the solution of instance \( I \).
An algorithm is a finite set of instructions for performing basic operations on an input to produce an output.

An algorithm $A$ solves a problem $P$ if given a representation of each instance $I$ as input it supplies as output the solution of instance $I$.

Instances can have more than one representation.
Algorithms

- An **algorithm** is a finite set of instructions for performing basic operations on an **input** to produce an **output**.

- An algorithm $A$ solves a problem $P$ if given a **representation** of each instance $I$ as input it supplies as output the solution of instance $I$.

- Instances can have more than one representation.
  - A graph $G = (V, E)$ can be represented as an adjacency matrix, incidence matrix, adjacency list, etc.
Running time

Question: How to measure the efficiency of an algorithm?

Answer: Running time - count of the number of elementary operations needed to run the algorithm.

Key insight: the number of operations depends on the difficulty of the instance and the size of the input.

Worst case difficulty: consider run times of “hard” instances. Express run time as a function of the amount of “memory” needed to represent the instance!
Running time

- **Question**: How to measure the efficiency of an algorithm $A$?
Running time

- **Question**: How to measure the efficiency of an algorithm $A$?
- **Answer**: Running time - count of the number of elementary operations needed to run the algorithm.
Running time

- **Question:** How to measure the efficiency of an algorithm $A$?
- **Answer:** Running time - count of the number of elementary operations needed to run the algorithm.

- **Key insight:** the number of operations depends on the difficulty of the instance and the size of the input.
Question: How to measure the efficiency of an algorithm $A$?

Answer: Running time - count of the number of elementary operations needed to run the algorithm.

Key insight: the number of operations depends on the difficulty of the instance and the size of the input.
  - Worst case difficulty: consider run times of “hard” instances.
Question: How to measure the efficiency of an algorithm $A$?

Answer: Running time - count of the number of elementary operations needed to run the algorithm.

Key insight: the number of operations depends on the difficulty of the instance and the size of the input.

- Worst case difficulty: consider run times of “hard” instances.
- Express run time as a function of the amount of “memory” needed to represent the instance!
The size of an instance depends on its representation. The encoding size of a representation is the number of binary digits (bits) needed to encode it into memory. Some examples of binary encoding sizes:

- $n = 118$.
- $n = 64 + 32 + 16 + 4 + 2 = 110110$ (in binary)
- That is, $|n| = 7$ bits
- More generally, $n \in \mathbb{Z}$ can be encoded in around $\log_2 n$ bits.
- The set $S = \{0, \ldots, n\}$ can also be encoded in around $\log_2 n$ bits.
The size of an instance depends on its representation.
The size of an instance depends on its representation.

The encoding size of a representation is the number of binary digits (bits) needed to encode it into memory.
The size of an instance depends on its representation.

The **encoding size** of a representation is the number of **binary digits** (bits) needed to encode it into memory.

Some examples of binary encoding sizes:

1. $n = 118$. 

$2^{118}$
The size of an instance depends on its representation.
The **encoding size** of a representation is the number of binary digits (bits) needed to encode it into memory.

Some examples of binary encoding sizes:

1. \( n = 118 \).

\[ n = 64 + 32 + 16 + 4 + 2 \]
The size of an instance depends on its representation.

The **encoding size** of a representation is the number of binary digits (bits) needed to encode it into memory.

Some examples of binary encoding sizes:

1. \( n = 118 \).

\[
\begin{align*}
n &= 64 + 32 + 16 + 4 + 2 \\
&= 110110 \text{ (in binary)}
\end{align*}
\]
The size of an instance depends on its representation. The encoding size of a representation is the number of binary digits (bits) needed to encode it into memory.

Some examples of binary encoding sizes:

1. \( n = 118 \).

\[
\begin{align*}
  n &= 64 + 32 + 16 + 4 + 2 \\
  &= 110110 \text{ (in binary)}
\end{align*}
\]

That is, \(|n| = 7\) bits
The size of an instance depends on its representation.

The encoding size of a representation is the number of binary digits (bits) needed to encode it into memory.

Some examples of binary encoding sizes:

1. $n = 118.$

$$n = 64 + 32 + 16 + 4 + 2$$
$$= 110110 \text{ (in binary)}$$

That is, $|n| = 7$ bits

More generally, $n \in \mathbb{Z}$ can be encoded in around $\log_2 n$ bits.
The size of an instance depends on its representation.

The **encoding size** of a representation is the number of binary digits (bits) needed to encode it into memory.

Some examples of binary encoding sizes:

1. \( n = 118 \).

\[
\begin{align*}
  n &= 64 + 32 + 16 + 4 + 2 \\
  &= 110110 \text{ (in binary)}
\end{align*}
\]

That is, \(|n| = 7\) bits.

More generally, \( n \in \mathbb{Z} \) can be encoded in around \( \log_2 n \) bits.

2. The set \( S = \{0, \ldots, n\} \) can also be encoded in around \( \log_2 n \) bits.
If the running time of an algorithm is bounded by a polynomial function of the input size, then we say the algorithm runs in polynomial time.

For example: ordering a finite list $L$ of numbers. Input size of the normal form representation of $L$ is the size $n$. Silly algorithm requires around $(n^2)$ comparisons and re-ordering of the lists.
If the running time of an algorithm is bounded by a polynomial function of the input size, then we say the algorithm runs in polynomial time.
If the running time of an algorithm is bounded by a polynomial function of the input size, then we say the algorithm runs in \textit{polynomial time}.

For example: ordering a finite list $\mathcal{L}$ of numbers
- Input size of the normal form representation of $\mathcal{L}$ is the size $n$. 

Polynomial time algorithms

- If the running time of an algorithm is bounded by a polynomial function of the input size, then we say the algorithm runs in **polynomial time**.

For example: ordering a finite list $\mathcal{L}$ of numbers

- Input size of the normal form representation of $\mathcal{L}$ is the size $n$.
- Silly algorithm requires around $\frac{n^2}{2}$ comparisons and re-ordering of the lists.
What do you mean is HARD TO COMPUTE X ??

Figure: I tried to compute X, I can’t do it, therefore it must be hard!
**Figure:** I can’t compute $X$, but if I could do it, the problems of all these people would be solved too! therefore it must be hard!

#P-complete problems is a family of COUNTING problems, if one finds a fast solution for one, you find it for all the members of the family!
Thank you