The Triangle Algorithm: An Algorithmic Separation Theorem and its Applications

Bahman Kalantari

Department of Computer Science, Rutgers University, New Brunswick, New Jersey, USA

DIMACS Workshop On Distance Geometry

July 26, 2016

The Convex Hull Membership Problem (CHMP)

The Convex Hull Membership Problem (CHMP)

Definition

Given a subset $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$, and $p \in \mathbb{R}^m$, either give a certificate that proves $p \in conv(S)$, or one that proves $p \notin conv(S)$.

The Convex Hull Membership Problem (CHMP)

Definition

Given a subset $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$, and $p \in \mathbb{R}^m$, either give a certificate that proves $p \in conv(S)$, or one that proves $p \notin conv(S)$.

Fact

$$p \in conv(S) \iff p = \sum_{i=1}^{n} \alpha_i v_i, \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i \geq 0.$$

The Convex Hull Membership Problem (CHMP)

Definition

Given a subset $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$, and $p \in \mathbb{R}^m$, either give a certificate that proves $p \in conv(S)$, or one that proves $p \notin conv(S)$.

Fact

$$p \in conv(S) \iff p = \sum_{i=1}^{n} \alpha_i v_i, \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i \geq 0.$$

Remark

The Convex Hull Membership Problem (CHMP)

Definition

Given a subset $S = \{v_1, \dots, v_n\} \subset \mathbb{R}^m$, and $p \in \mathbb{R}^m$, either give a certificate that proves $p \in conv(S)$, or one that proves $p \notin conv(S)$.

Fact

$$p \in conv(S) \iff p = \sum_{i=1}^{n} \alpha_i v_i, \quad \sum_{i=1}^{n} \alpha_i = 1, \quad \alpha_i \geq 0.$$

Remark

When $p \notin conv(S)$ a certificate is a separating hyperplane.

Definition

(**Homogeneous CHMP** (H-CHMP))

Homogeneous and Approximate Version of CHMP

Definition

(**Homogeneous CHMP** (H-CHMP))
Given an $m \times n$ matrix A, either find x satisfying

$$Ax = 0, \quad e^T x = 1, \quad x \geq 0,$$

or prove unsolvable. (test if $0 \in conv(A)$)

Homogeneous and Approximate Version of CHMP

Definition

(**Homogeneous CHMP** (H-CHMP))
Given an $m \times n$ matrix A, either find x satisfying

$$Ax = 0, \quad e^T x = 1, \quad x \geq 0,$$

or prove unsolvable. (test if $0 \in conv(A)$)

Definition

(ε-approximate version of CHMP)

Homogeneous and Approximate Version of CHMP

Definition

(**Homogeneous CHMP** (H-CHMP))
Given an $m \times n$ matrix A, either find x satisfying

$$Ax = 0, \quad e^T x = 1, \quad x \geq 0,$$

or prove unsolvable. (test if $0 \in conv(A)$)

Definition

(ε-approximate version of CHMP)
Given $\varepsilon \in (0, 1)$, either compute $p_\varepsilon \in conv(S)$ such that:

Homogeneous and Approximate Version of CHMP

Definition

(**Homogeneous CHMP** (H-CHMP))
Given an $m \times n$ matrix A, either find x satisfying

$$Ax = 0, \quad e^T x = 1, \quad x \geq 0,$$

or prove unsolvable. (test if $0 \in conv(A)$)

Definition

(ε-approximate version of CHMP)
Given $\varepsilon \in (0, 1)$, either compute $p_\varepsilon \in conv(S)$ such that:

$$d(p_\varepsilon, p) \leq \varepsilon \cdot R, \quad R = \max\{d(p, v_1), \dots, d(p, v_n)\};$$

Homogeneous and Approximate Version of CHMP

Definition

(**Homogeneous CHMP** (H-CHMP))
Given an $m \times n$ matrix A, either find x satisfying

$$Ax = 0, \quad e^T x = 1, \quad x \geq 0,$$

or prove unsolvable. (test if $0 \in conv(A)$)

Definition

(ε-approximate version of CHMP)
Given $\varepsilon \in (0,1)$, either compute $p_\varepsilon \in conv(S)$ such that:

$$d(p_\varepsilon, p) \leq \varepsilon \cdot R, \quad R = \max\{d(p, v_1), \dots, d(p, v_n)\};$$

or prove $p \notin conv(S)$. ($d(u,v) = \|u - v\|$, Euclidean distance)

Significance of CHMP and H-CHMP

Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.

Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:

Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:

Gordan's Theorem (1873) (preceded Farkas Lemma),

Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:

Gordan's Theorem (1873) (preceded Farkas Lemma),
Diagonal Scaling Dualities,

Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:

Gordan's Theorem (1873) (preceded Farkas Lemma),
Diagonal Scaling Dualities,
Distance Dualities (to be described).

Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:

Gordan's Theorem (1873) (preceded Farkas Lemma),
Diagonal Scaling Dualities,
Distance Dualities (to be described).

- In fact these are most fundamental problems in linear programming.

Significance of CHMP and H-CHMP

- Applications in approximation theory, machine learning, statistics, etc.
- It has given rise to significant dualities and algorithms:

Gordan's Theorem (1873) (preceded Farkas Lemma),
Diagonal Scaling Dualities,
Distance Dualities (to be described).

- In fact these are most fundamental problems in linear programming.
- Historically speaking, the first two polynomial-time LP algorithm happened to be (implicitly) designed for solve H-CHMP:

Significance of CHMP and H-CHMP

Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

$$\text{Is} \quad \min\{c^T x : Ax = 0, \quad e^T x = 1, x \geq 0\} = 0? \quad \text{(H-CHMP)}$$

Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

$$\text{Is} \quad \min\{c^T x : Ax = 0, \quad e^T x = 1, x \geq 0\} = 0? \quad \text{(H-CHMP)}$$

- Khachiyan ellipsoid algorithm (1979) solves:

Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

$$\text{Is} \quad \min\{c^T x : Ax = 0, \quad e^T x = 1, x \geq 0\} = 0? \quad \text{(H-CHMP)}$$

- Khachiyan ellipsoid algorithm (1979) solves:

$$\text{Is } \; Ax < b \; \text{ feasible?}$$

$$\text{Is } \; 0 \in conv(B)?, \quad B = \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}. \quad \text{(dual of } \; Ax < b\text{)}$$

Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

$$\text{Is} \quad \min\{c^T x : Ax = 0, \quad e^T x = 1, x \geq 0\} = 0? \quad \text{(H-CHMP)}$$

- Khachiyan ellipsoid algorithm (1979) solves:

$$\text{Is} \;\; Ax < b \;\; \text{feasible?}$$

$$\text{Is} \;\; 0 \in conv(B)?, \quad B = \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}. \quad (\text{dual of} \;\; Ax < b)$$

- Khachiyan-K. matrix scaling algorithm (1992):

Significance of CHMP and H-CHMP

- Karmarkar projective algorithm (1984) solves:

$$\text{Is} \quad \min\{c^T x : Ax = 0, \quad e^T x = 1, x \geq 0\} = 0? \quad \text{(H-CHMP)}$$

- Khachiyan ellipsoid algorithm (1979) solves:

$$\text{Is } \; Ax < b \; \text{ feasible?}$$

$$\text{Is } \; 0 \in conv(B)?, \quad B = \begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix}. \quad \text{(dual of } \; Ax < b\text{)}$$

- Khachiyan-K. matrix scaling algorithm (1992): Given an $n \times n$ symmetric psd matrix A, test the solvability of the following nonlinear dual to H-CHMP ($0 \in conv(A)$?):

$$DADe = e, \quad D = \mathrm{diag}(d_1, \cdots, d_n), \quad d_i > 0, \quad e = (1, \ldots, 1)^T.$$

Significance of CHMP and H-CHMP

Significance of CHMP and H-CHMP

- Indeed a generalization of H-CHMP, called *Homogeneous Programming*, is a special but significant conic programming problem: Given a homogeneous function $\phi(x)$,

Significance of CHMP and H-CHMP

• Indeed a generalization of H-CHMP, called *Homogeneous Programming*, is a special but significant conic programming problem: Given a homogeneous function $\phi(x)$,

$$\text{Is} \quad \phi(x) = 0?, x \neq 0, x \in K, \quad \text{a pointed cone.}$$

Significance of CHMP and H-CHMP

• Indeed a generalization of H-CHMP, called *Homogeneous Programming*, is a special but significant conic programming problem: Given a homogeneous function $\phi(x)$,

$$\text{Is} \quad \phi(x) = 0?, x \neq 0, x \in K, \quad \text{a pointed cone.}$$

The matrix scaling equation $DADe = e$ holds with appropriate interrelation of D and A and e, dependent on ϕ and K.

Significance of CHMP and H-CHMP

• Indeed a generalization of H-CHMP, called *Homogeneous Programming*, is a special but significant conic programming problem: Given a homogeneous function $\phi(x)$,

$$\text{Is} \quad \phi(x) = 0?, x \neq 0, x \in K, \quad \text{a pointed cone.}$$

The matrix scaling equation $DADe = e$ holds with appropriate interrelation of D and A and e, dependent on ϕ and K.

Moreover, algorithmic applications of these for semidefinite programming and self-concordant programming have been analyzed, e.g. "Semidefinite programming and matrix scaling over the semidefinite cone," *Linear Algebra and its Applications*, 2003, B.K.

Triangle Algorithm : A Geometric Algorithm for CHMP

Triangle Algorithm : A Geometric Algorithm for CHMP

Triangle Algorithm ($S = \{v_1, \ldots, v_n\}$, p)

Triangle Algorithm : A Geometric Algorithm for CHMP

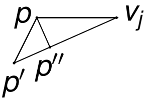
Triangle Algorithm ($S = \{v_1, \ldots, v_n\}$, p)

- **Step 1.** Given *iterate* $p' = \sum_{i=1}^{n} \alpha_i v_i \in conv(S)$, check if there exists a *pivot* : $v_j \in S$ s.t. $d(p', v_j) \geq d(p, v_j)$.

Triangle Algorithm : A Geometric Algorithm for CHMP

Triangle Algorithm ($S = \{v_1, \ldots, v_n\}$, p)

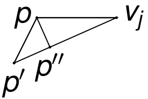
- **Step 1.** Given *iterate* $p' = \sum_{i=1}^{n} \alpha_i v_i \in conv(S)$, check if there exists a *pivot* : $v_j \in S$ s.t. $d(p', v_j) \geq d(p, v_j)$.



Triangle Algorithm : A Geometric Algorithm for CHMP

Triangle Algorithm ($S = \{v_1, \ldots, v_n\}$, p)

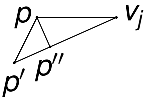
- **Step 1.** Given *iterate* $p' = \sum_{i=1}^{n} \alpha_i v_i \in conv(S)$, check if there exists a *pivot* : $v_j \in S$ s.t. $d(p', v_j) \geq d(p, v_j)$.



If no pivot exists, then p' is a *witness*. Stop.

Triangle Algorithm ($S = \{v_1, \dots, v_n\}$, p)

- **Step 1.** Given *iterate* $p' = \sum_{i=1}^{n} \alpha_i v_i \in conv(S)$, check if there exists a *pivot* : $v_j \in S$ s.t. $d(p', v_j) \geq d(p, v_j)$.



  If no pivot exists, then p' is a *witness*. Stop.

- **Step 2.** Otherwise, compute $p'' = nearest(p; p'v)$:

$$p'' = (1-\alpha)p' + \alpha v_j = \sum_{i=1}^{n} \alpha_i' v_i, \quad \alpha = \frac{(p - p')^T (v_j - p')}{d^2(v_j, p')},$$

$$\alpha_j' = (1-\alpha)\alpha_j + \alpha, \quad \alpha_i = (1-\alpha)\alpha_i, \quad \forall i \neq j.$$

  Replace p' with p'' and Go to Step 1.

Example of Triangle Algorithm for a Triangle

Example of Triangle Algorithm for a Triangle

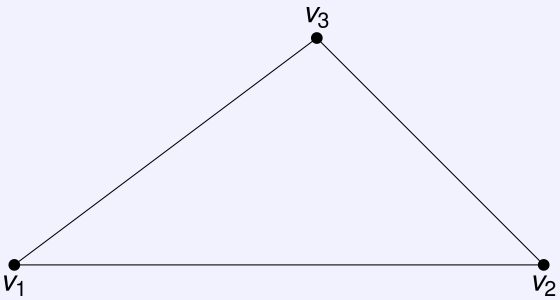


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for a Triangle

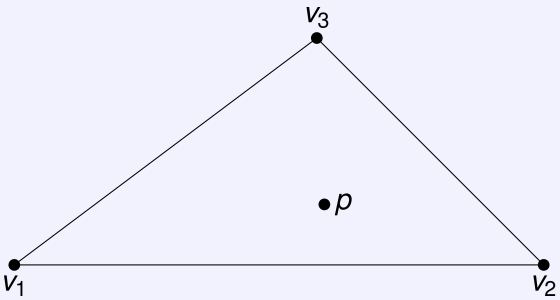


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for a Triangle

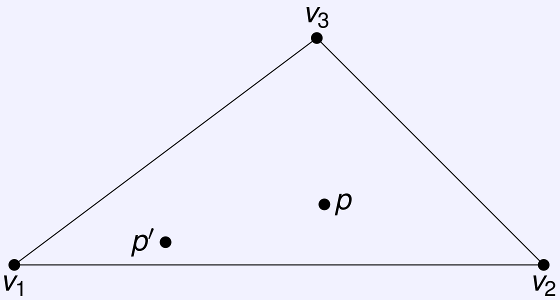


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for a Triangle

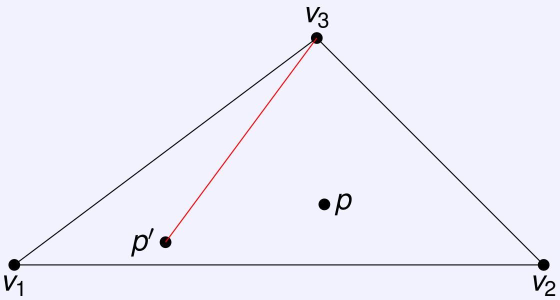


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for a Triangle

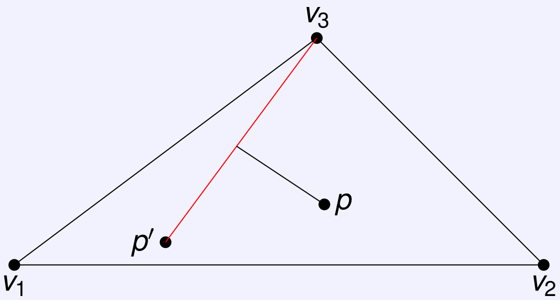


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for a Triangle

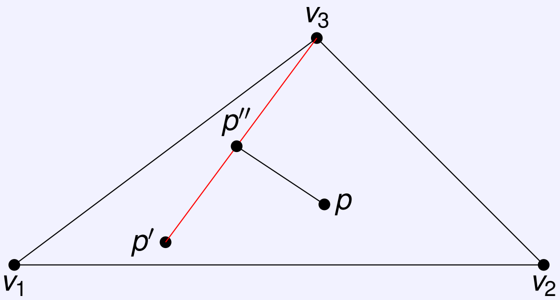


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for a Triangle

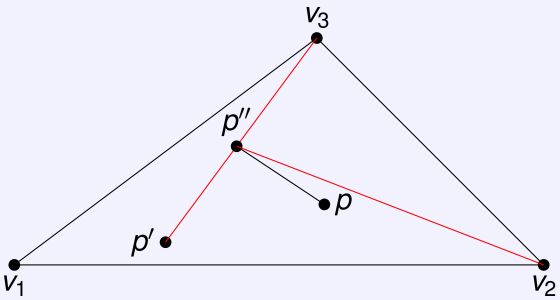


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for a Triangle

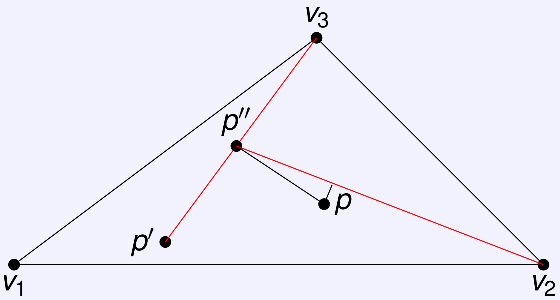


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for a Triangle

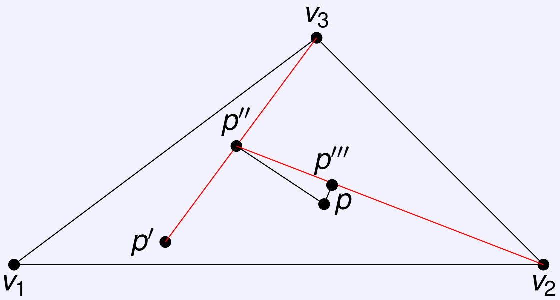


Figure: Triangle Algorithm for testing if $p \in conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for A Triangle

Example of Triangle Algorithm for A Triangle

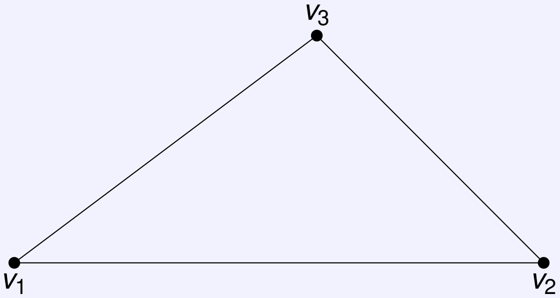


Figure: A case of $p \notin conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for A Triangle

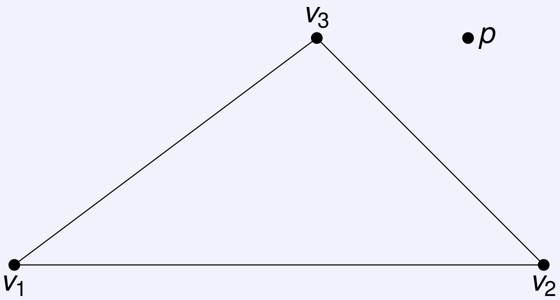


Figure: A case of $p \notin conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for A Triangle

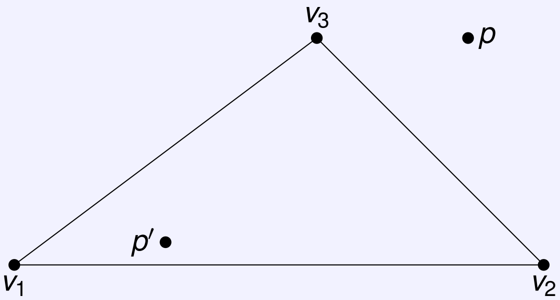


Figure: A case of $p \notin conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for A Triangle

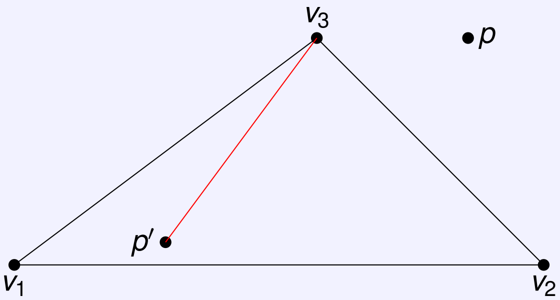


Figure: A case of $p \notin conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for A Triangle

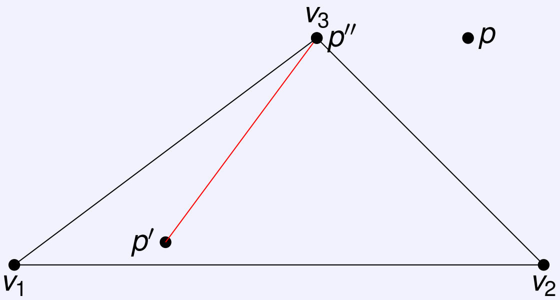


Figure: A case of $p \notin conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for A Triangle

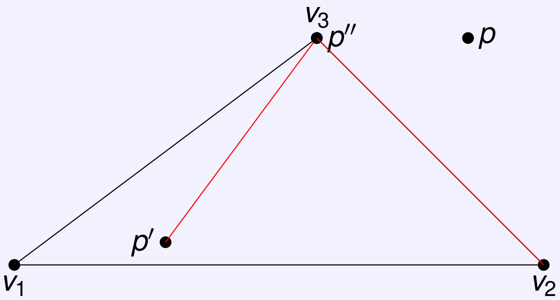


Figure: A case of $p \notin conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for A Triangle

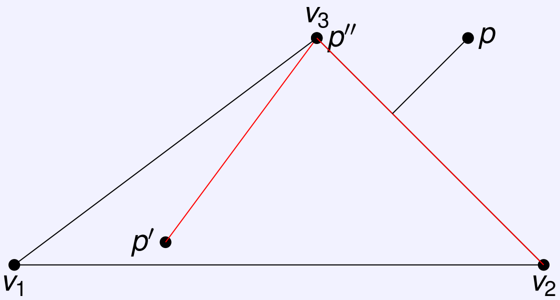


Figure: A case of $p \notin conv(\{v_1, v_2, v_3\})$.

Example of Triangle Algorithm for A Triangle

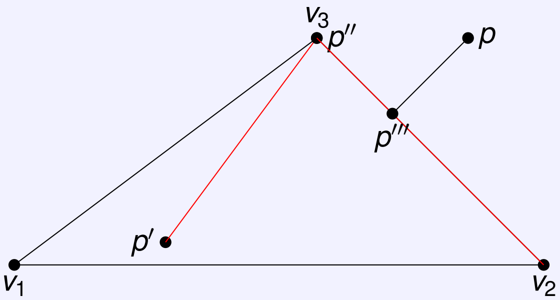


Figure: A case of $p \notin conv(\{v_1, v_2, v_3\})$.

Geometry of Triangle Algorithm

Geometry of Triangle Algorithm

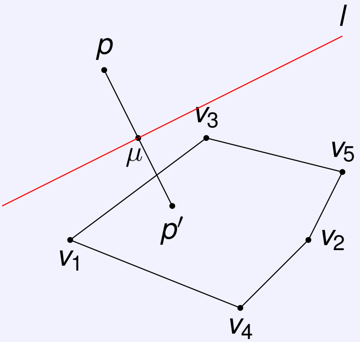


Figure: When orthogonal bisector of pp' separate p from $conv(S)$ (left) and when it does not.

Geometry of Triangle Algorithm

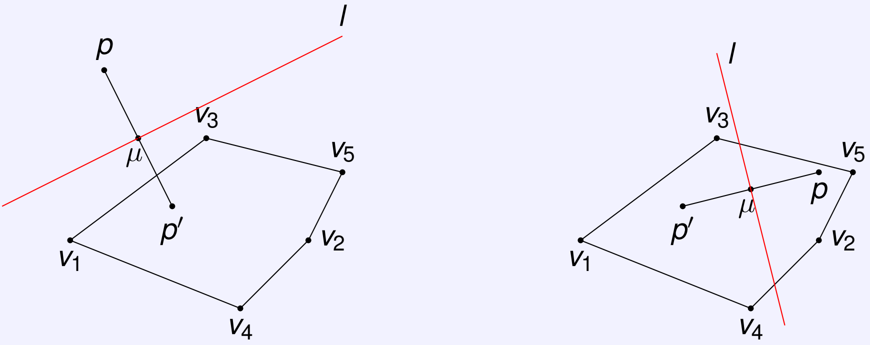


Figure: When orthogonal bisector of pp' separate p from $conv(S)$ (left) and when it does not.

Geometry of Triangle Algorithm

Geometry of Triangle Algorithm

Theorem

(Distance Duality)

Theorem

(Distance Duality)

Precisely one of the two conditions is satisfied:

Geometry of Triangle Algorithm

Theorem

(Distance Duality)
Precisely one of the two conditions is satisfied:
(i): For each $p' \in conv(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot)

Geometry of Triangle Algorithm

Theorem

(Distance Duality)

Precisely one of the two conditions is satisfied:

(i): For each $p' \in conv(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot)

(ii): There exists $p' \in conv(S)$ such that $d(p', v) < d(p, v)$, for all $v \in S$ (p' a witness).

Geometry of Triangle Algorithm

Theorem

(Distance Duality)
Precisely one of the two conditions is satisfied:
(i): For each $p' \in conv(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot)
(ii): There exists $p' \in conv(S)$ such that $d(p', v) < d(p, v)$, for all $v \in S$ (p' a witness).
Furthermore, (i) is valid if and only if $p \in conv(S)$.

Geometry of Triangle Algorithm

Theorem

(Distance Duality)
Precisely one of the two conditions is satisfied:
(i): For each $p' \in conv(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot)
(ii): There exists $p' \in conv(S)$ such that $d(p', v) < d(p, v)$, for all $v \in S$ (p' a witness).
Furthermore, (i) is valid if and only if $p \in conv(S)$.
Equivalently, (ii) is valid if and only if $p \notin conv(S)$.

Geometry of Triangle Algorithm

Theorem

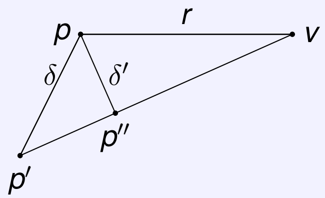
(Distance Duality)
Precisely one of the two conditions is satisfied:
(i): For each $p' \in conv(S)$, there exists $v \in S$ such that $d(p', v) \geq d(p, v)$ (v a pivot)
(ii): There exists $p' \in conv(S)$ such that $d(p', v) < d(p, v)$, for all $v \in S$ (p' a witness).
Furthermore, (i) is valid if and only if $p \in conv(S)$.
Equivalently, (ii) is valid if and only if $p \notin conv(S)$.

Remark

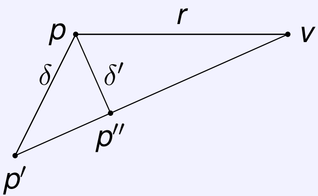
H.W. Kuhn (1967), proves this in the Euclidean plane making use of several results, including Ville's Lemma. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986).

Iterative Reduction of Error Gap

Iterative Reduction of Error Gap



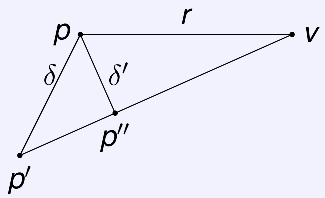
Iterative Reduction of Error Gap



Theorem

Given two consecutive iterates p', p'', corresponding to the triangle $\triangle pp'v$ with v a pivot, let $\delta = d(p', p)$, $\delta' = d(p'', p)$, and $r = d(p, v)$.

Iterative Reduction of Error Gap



Theorem

Given two consecutive iterates p', p'', *corresponding to the triangle* $\triangle pp'v$ *with* v *a pivot, let* $\delta = d(p', p)$, $\delta' = d(p'', p)$, *and* $r = d(p, v)$. *Then, if* $\delta \leq r$,

$$\delta' \leq \delta \sqrt{1 - \frac{\delta^2}{4r^2}}.$$

Complexity of Triangle Algorithm: First Bound

Theorem

(i) Suppose $p \in conv(S)$. *Given* $\varepsilon > 0$, *the number of iterations to compute a point* p_ε *in* $conv(S)$ *so that* $d(p, p_\varepsilon) \leq \varepsilon R$, $R = \max\{d(p, v_1), \ldots, d(p, v_n)\}$ *is*

$$O\left(\frac{1}{\varepsilon^2}\right).$$

Complexity of Triangle Algorithm: First Bound

Theorem

(i) Suppose $p \in conv(S)$. Given $\varepsilon > 0$, the number of iterations to compute a point p_ε in $conv(S)$ so that $d(p, p_\varepsilon) \leq \varepsilon R$, $R = \max\{d(p, v_1), \ldots, d(p, v_n)\}$ is

$$O\left(\frac{1}{\varepsilon^2}\right).$$

(ii) Suppose $p \notin conv(S)$. The number of iterations to compute a witness p' in $conv(S)$ is

$$O\left(\frac{R^2}{\Delta^2}\right), \quad \Delta = \min\left\{d(x, p) : \quad x \in conv(S)\right\}.$$

Remarks on the Triangle Algorithm

Remarks on the Triangle Algorithm

Remark

In straightforward implementation, worst-case complexity in each iteration is $O(mn)$ arithmetic operations.

Remarks on the Triangle Algorithm

Remark

In straightforward implementation, worst-case complexity in each iteration is $O(mn)$ arithmetic operations.

Remark

With a preprocessing time of $O(mn^2)$, each iteration can be implemented in $O(m+n)$ arithmetic operations.

Remarks on the Triangle Algorithm

Remark

In straightforward implementation, worst-case complexity in each iteration is $O(mn)$ arithmetic operations.

Remark

With a preprocessing time of $O(mn^2)$, each iteration can be implemented in $O(m+n)$ arithmetic operations.

Remark

To find pivot Triangle Algorithm does not require taking square-roots:

$$d(p', v) \geq d(p, v) \iff \|p'\|^2 - \|p\|^2 \geq 2v^T(p' - p).$$

Remarks on Other Algorithms for Solving CHMP

Remarks on Other Algorithms for Solving CHMP

Remark

- *Simplex Method solves CHMP as Phase I.*

Remarks on Other Algorithms for Solving CHMP

Remark

- *Simplex Method solves CHMP as Phase I.*
- *Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.*

Remarks on Other Algorithms for Solving CHMP

Remark

- *Simplex Method solves CHMP as Phase I.*
- *Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.*
- *Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert's algorithm.*

Remarks on Other Algorithms for Solving CHMP

Remark

- *Simplex Method solves CHMP as Phase I.*
- *Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.*
- *Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert's algorithm. Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.*

Remarks on Other Algorithms for Solving CHMP

Remark

- *Simplex Method solves CHMP as Phase I.*
- *Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.*
- *Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert's algorithm. Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.*
- *So-called fast gradient method of Nesterov can also be applied, an $O(1/\varepsilon)$ iteration algorithm, complexity of each iteration is $O(mn)$.*

Remarks on Other Algorithms for Solving CHMP

Remark

- *Simplex Method solves CHMP as Phase I.*
- *Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.*
- *Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert's algorithm. Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.*
- *So-called fast gradient method of Nesterov can also be applied, an $O(1/\varepsilon)$ iteration algorithm, complexity of each iteration is $O(mn)$.*
- *Worst-case complexity of each iteration of Triangle Algorithm is $O(mn)$. However, even without preprocessing, often, each iteration requires only $O(m+n)$.*

Remarks on Other Algorithms for Solving CHMP

Remark

- *Simplex Method solves CHMP as Phase I.*
- *Sparse greedy approximation solves CHMP by conversion into a convex quadratic minimization over a simplex.*
- *Sparse greedy approximation is equivalent to Frank-Wolf, also Gilbert's algorithm. Motivation behind their iterative steps is algebraic - Triangle Algorithm is motivated by geometric properties.*
- *So-called fast gradient method of Nesterov can also be applied, an $O(1/\varepsilon)$ iteration algorithm, complexity of each iteration is $O(mn)$.*
- *Worst-case complexity of each iteration of Triangle Algorithm is $O(mn)$. However, even without preprocessing, often, each iteration requires only $O(m+n)$.*
- *Triangle Algorithm could outperform these due to distance duality, simplicity and degrees of freedom it offers.*

Experimental Results with Triangle Algorithm

Experimental Results with Triangle Algorithm

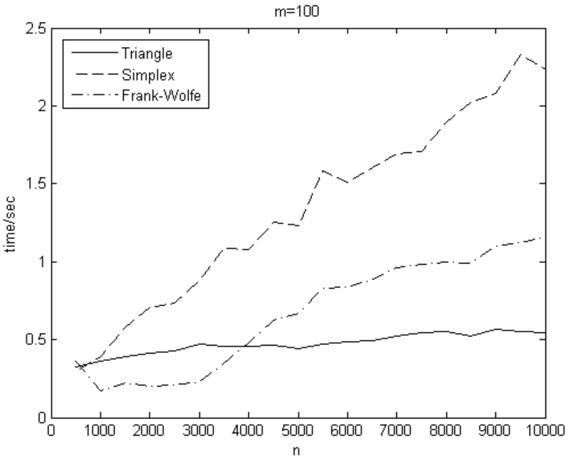


Figure: Running time comparison as n grows

Experimental Results with Triangle Algorithm

Experimental Results with Triangle Algorithm

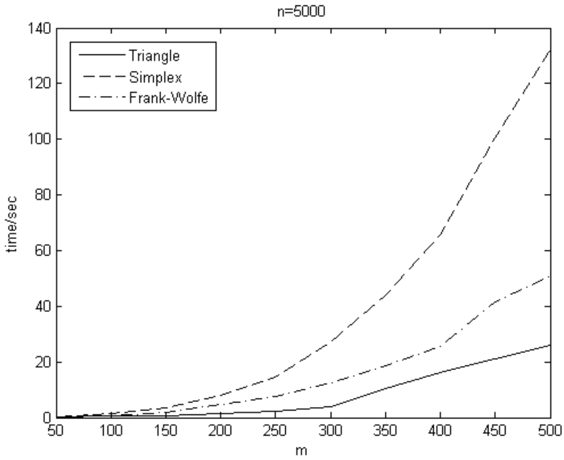


Figure: Running time comparison as m grows

Experimental Results with Triangle Algorithm

Experimental Results with Triangle Algorithm

As the number of points n grow, the running time of the Simplex and Frank-Wolfe methods increase while the Triangle Algorithm performs very well with only a slight increase in the running time.

Experimental Results with Triangle Algorithm

As the number of points n grow, the running time of the Simplex and Frank-Wolfe methods increase while the Triangle Algorithm performs very well with only a slight increase in the running time.

One explanation is the fact that the Triangle Algorithm does not need to make use of all the n points and thus spends less time than the simplex method and Frank-Wolfe in each iteration. Another is that by virtue of selecting a pivot it makes good reductions in each iteration.

Experimental Results with Triangle Algorithm

As the number of points *n* grow, the running time of the Simplex and Frank-Wolfe methods increase while the Triangle Algorithm performs very well with only a slight increase in the running time.

One explanation is the fact that the Triangle Algorithm does not need to make use of all the *n* points and thus spends less time than the simplex method and Frank-Wolfe in each iteration. Another is that by virtue of selecting a pivot it makes good reductions in each iteration.

n	# of points visited per iteration	iterations
500	185	459.6
1000	228.26	479.6
3000	240.37	540.4
5000	242.22	541.6
10000	254.84	535.4

Table: The performance of Triangle algorithm when m=100

Properties and Characterizations of Witnesses: Separation Property

Properties and Characterizations of Witnesses: Separation Property

Definition

Let W_p be the set of all witnesses, i.e. points $p' \in conv(S)$ such that

$$d(p', v_i) < d(p, v_i), \quad \forall i = 1, \ldots, n.$$

Properties and Characterizations of Witnesses: Separation Property

Definition

Let W_p be the set of all witnesses, i.e. points $p' \in conv(S)$ such that

$$d(p', v_i) < d(p, v_i), \quad \forall i = 1, \dots, n.$$

Theorem

If $p' \in W_p$ the orthogonal bisecting hyperplane of the line segment pp' separates p from $conv(S)$.

Properties and Characterizations of Witnesses: Approximation of Distance to Convex Hull

Properties and Characterizations of Witnesses: Approximation of Distance to Convex Hull

Corollary

Suppose $p \notin conv(S) = conv(\{v_1, \ldots, v_n\})$.

Properties and Characterizations of Witnesses: Approximation of Distance to Convex Hull

Corollary

Suppose $p \notin conv(S) = conv(\{v_1, \ldots, v_n\})$.
Let

$$\Delta = d(p, conv(S)) = \min\{d(p, x) : \quad x \in conv(S)\}.$$

Properties and Characterizations of Witnesses: Approximation of Distance to Convex Hull

Corollary

Suppose $p \notin conv(S) = conv(\{v_1, \ldots, v_n\})$. *Let*

$$\Delta = d(p, conv(S)) = \min\{d(p,x) : \quad x \in conv(S)\}.$$

Then any witness $p' \in W_p$ *gives an estimate of* Δ *to within a factor of two. More precisely,*

Properties and Characterizations of Witnesses: Approximation of Distance to Convex Hull

Corollary

Suppose $p \notin conv(S) = conv(\{v_1, \ldots, v_n\})$*.*
Let

$$\Delta = d(p, conv(S)) = \min\{d(p,x) : \quad x \in conv(S)\}.$$

Then any witness $p' \in W_p$ *gives an estimate of* Δ *to within a factor of two. More precisely,*

$$\frac{1}{2} d(p,p') \leq \Delta \leq d(p,p').$$

Properties and Characterizations of Witnesses: Intersection Ball Property

Properties and Characterizations of Witnesses: Intersection Ball Property

Corollary

Given $S = \{v_1, \dots, v_n\}$ and p all in $\mathbb{R}^m$, consider the set of open balls B_i balls centered at v_i with radius $d(p, v_i)$, $i = 1, \dots, n$.
Then $p \in conv(S)$ if and only if $(\cap_{i=1}^n B_i) \cap conv(S) = \emptyset$.
Equivalently, $p \in conv(S)$ if and only if $(\cap_{i=1}^n \overline{B}_i) \cap conv(S) = \emptyset$.

A Case with No Witness: $p \in conv(S)$

A Case with No Witness: $p \in conv(S)$

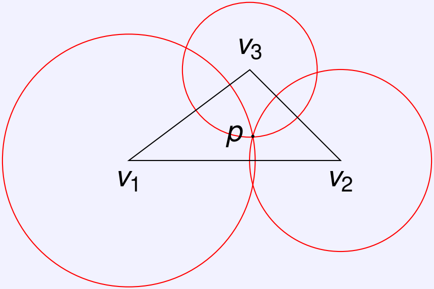


Figure: No witnesses: $p \in conv(S)$. The three discs intersect only at p.

Some Cases with Witnesses: $p \notin conv(S)$

Some Cases with Witnesses: $p \notin conv(S)$

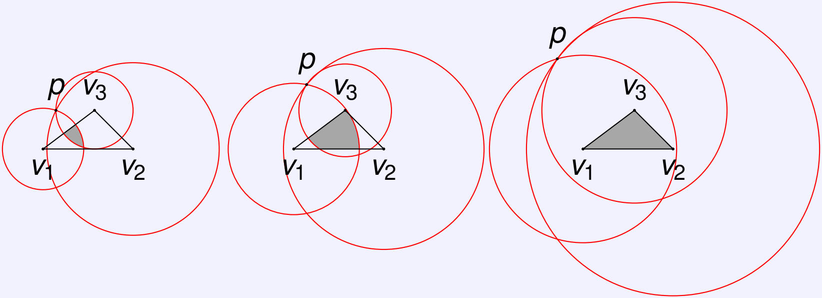


Figure: Examples with $W_p \neq \emptyset$, $p \notin conv(S)$. W_p is interior of gray areas: For any $p' \in W_p$ the bisector of pp' separates p from $conv(S)$.

Strict Distance Duality

Strict Distance Duality

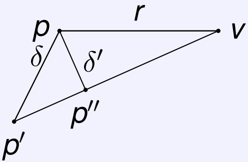
Definition

Given $p' \in conv(S)$, we say $v \in S$ is a *strict pivot* if $\angle p'pv \geq \pi/2$.

Strict Distance Duality

Definition

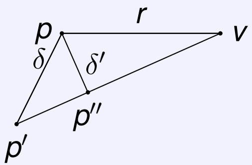
Given $p' \in conv(S)$, we say $v \in S$ is a *strict pivot* if $\angle p'pv \geq \pi/2$.



Strict Distance Duality

Definition

Given $p' \in conv(S)$, we say $v \in S$ is a *strict pivot* if $\angle p'pv \geq \pi/2$.



Theorem

(Strict Distance Duality) Assume $p \notin S$. Then $p \in conv(S)$ if and only if for each $p' \in conv(S)$ there exists a strict pivot.

Complexity of Triangle Algorithm: Second Bound

Complexity of Triangle Algorithm: Second Bound

Theorem

Complexity of Triangle Algorithm: Second Bound

Theorem

Assume p lies at the center of a ball of radius ρ in the relative interior of conv(S), and Triangle Algorithm uses strict pivot in each iteration.

Complexity of Triangle Algorithm: Second Bound

Theorem

Assume p lies at the center of a ball of radius ρ in the relative interior of $conv(S)$, and Triangle Algorithm uses strict pivot in each iteration. The number of iterations to compute $p_\varepsilon \in conv(S)$ such that

$$d(p, p_\varepsilon) < \varepsilon R, \quad R = max\{d(p, v_i), i = 1, \ldots, n\}$$

satisfies

Complexity of Triangle Algorithm: Second Bound

Theorem

Assume p lies at the center of a ball of radius ρ in the relative interior of conv(S), and Triangle Algorithm uses strict pivot in each iteration. The number of iterations to compute $p_\varepsilon \in conv(S)$ such that

$$d(p, p_\varepsilon) < \varepsilon R, \quad R = max\{d(p, v_i), i = 1, \ldots, n\}$$

satisfies

$$O\left(\left(\frac{R}{\rho}\right)^2 \log\frac{1}{\varepsilon}\right).$$

Complexity of Triangle Algorithm: Third Bound

Complexity of Triangle Algorithm: Third Bound

Theorem

Complexity of Triangle Algorithm: Third Bound

Theorem

Given $\varepsilon \in (0,1)$, the number of iterations of the Triangle Algorithm to test if there exists $p_\varepsilon \in conv(S)$ such that $d(p, p_\varepsilon) < \varepsilon R$, $R = max\{d(p, v_i), i = 1, \dots, n\}$, is

Complexity of Triangle Algorithm: Third Bound

Theorem

Given $\varepsilon \in (0,1)$, the number of iterations of the Triangle Algorithm to test if there exists $p_\varepsilon \in conv(S)$ such that $d(p, p_\varepsilon) < \varepsilon R$, $R = max\{d(p, v_i), i = 1, \ldots, n\}$, is

$$O\left(\frac{1}{c} \ln \frac{1}{\varepsilon}\right), \tag{1}$$

Complexity of Triangle Algorithm: Third Bound

Theorem

Given $\varepsilon \in (0,1)$*, the number of iterations of the Triangle Algorithm to test if there exists* $p_\varepsilon \in conv(S)$ *such that* $d(p, p_\varepsilon) < \varepsilon R$*,* $R = max\{d(p, v_i), i = 1, \dots, n\}$*, is*

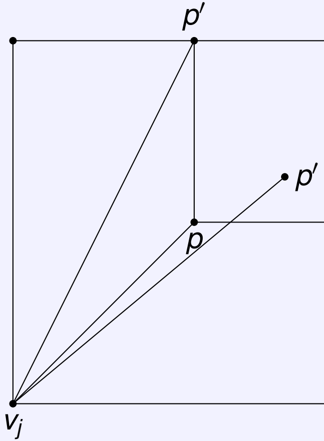
$$O\left(\frac{1}{c} \ln \frac{1}{\varepsilon}\right), \quad (1)$$

where c is the visibility factor, a constant satisfying the inequalities

$$\sin(pp'v') \leq \frac{1}{\sqrt{1+c}}, \quad c \geq \varepsilon^2, \quad (2)$$

over all the iterates p' *having corresponding pivot* v'*.*

Example



Strict Witness

Strict Witness

Definition

We say $p' \in conv(S)$ is a *strict witness* if there is no strict pivot at p'. Equivalently, p' is a strict witness if the orthogonal hyperplane to the line $p'p$ at p separates p from $conv(S)$. Denote the set of all strict witnesses by $\widehat{W}_p$.

Strict Witness

Definition

We say $p' \in conv(S)$ is a *strict witness* if there is no strict pivot at p'. Equivalently, p' is a strict witness if the orthogonal hyperplane to the line $p'p$ at p separates p from $conv(S)$. Denote the set of all strict witnesses by $\widehat{W}_p$.

$\widehat{W}_p$ contains W_p.

Strict Witness

Definition

We say $p' \in conv(S)$ is a *strict witness* if there is no strict pivot at p'. Equivalently, p' is a strict witness if the orthogonal hyperplane to the line $p'p$ at p separates p from $conv(S)$. Denote the set of all strict witnesses by $\widehat{W}_p$.

$\widehat{W}_p$ contains W_p.

Proposition

We have

$$\widehat{W}_p = \Big\{ x \in conv(S) : (x-p)^T(v_i - p) > 0, i = 1, \dots, n \Big\}.$$

Strict Witness

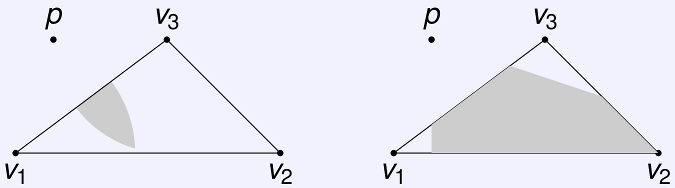


Figure: Witness set W_p (left) and Strict Witness set $\widehat{W}_p$ (right).

Solving Strict LP Feasibility Via Triangle Algorithm

Solving Strict LP Feasibility Via Triangle Algorithm

Test if $Ax < b$ is feasible, A is an $m \times n$ matrix.

Solving Strict LP Feasibility Via Triangle Algorithm

Test if $Ax < b$ is feasible, A is an $m \times n$ matrix.
(The problem Khachiyan considered in 1979).

Solving Strict LP Feasibility Via Triangle Algorithm

Test if $Ax < b$ is feasible, A is an $m \times n$ matrix.
(The problem Khachiyan considered in 1979).
$Ax < b$ is feasible if and only if the following CHMP is infeasible

$$\begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix} \begin{pmatrix} y \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \; y \geq 0, \; s \geq 0.$$

Solving Strict LP Feasibility Via Triangle Algorithm

Test if $Ax < b$ is feasible, A is an $m \times n$ matrix.
(The problem Khachiyan considered in 1979).
$Ax < b$ is feasible if and only if the following CHMP is infeasible

$$\begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix} \begin{pmatrix} y \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1,\ y \geq 0,\ s \geq 0.$$

Denote rows of A by a_i^T.

Solving Strict LP Feasibility Via Triangle Algorithm

Test if $Ax < b$ is feasible, A is an $m \times n$ matrix.
(The problem Khachiyan considered in 1979).
$Ax < b$ is feasible if and only if the following CHMP is infeasible

$$\begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix} \begin{pmatrix} y \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \; y \geq 0, \; s \geq 0.$$

Denote rows of A by a_i^T. Then columns of matrix in CHMP are $v_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix}$, $i = 1, \ldots, m$ and $v_{m+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, all in $\mathbb{R}^{n+1}$.

Solving Strict LP Feasibility Via Triangle Algorithm

Test if $Ax < b$ is feasible, A is an $m \times n$ matrix.
(The problem Khachiyan considered in 1979).
$Ax < b$ is feasible if and only if the following CHMP is infeasible

$$\begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix} \begin{pmatrix} y \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \; y \geq 0, \; s \geq 0.$$

Denote rows of A by a_i^T. Then columns of matrix in CHMP are $v_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix}$, $i = 1, \dots, m$ and $v_{m+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, all in $\mathbb{R}^{n+1}$.

Suppose triangle algorithm for CHMP gives a witness $p' = \begin{pmatrix} x \\ \alpha \end{pmatrix}$. Then,

Solving Strict LP Feasibility Via Triangle Algorithm

Test if $Ax < b$ is feasible, A is an $m \times n$ matrix.
(The problem Khachiyan considered in 1979).
$Ax < b$ is feasible if and only if the following CHMP is infeasible

$$\begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix} \begin{pmatrix} y \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \; y \geq 0, \; s \geq 0.$$

Denote rows of A by a_i^T. Then columns of matrix in CHMP are $v_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix}$, $i = 1, \ldots, m$ and $v_{m+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, all in $\mathbb{R}^{n+1}$.

Suppose triangle algorithm for CHMP gives a witness $p' = \begin{pmatrix} x \\ \alpha \end{pmatrix}$. Then,

$$A(-x/\alpha) < b.$$

Solving Strict LP Feasibility Via Triangle Algorithm

Test if $Ax < b$ is feasible, A is an $m \times n$ matrix.
(The problem Khachiyan considered in 1979).
$Ax < b$ is feasible if and only if the following CHMP is infeasible

$$\begin{pmatrix} A^T & 0 \\ b^T & 1 \end{pmatrix} \begin{pmatrix} y \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \sum_{i=1}^{m} y_i + s = 1, \; y \geq 0, \; s \geq 0.$$

Denote rows of A by a_i^T. Then columns of matrix in CHMP are $v_i = \begin{pmatrix} a_i \\ b_i \end{pmatrix}$, $i = 1, \ldots, m$ and $v_{m+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, all in $\mathbb{R}^{n+1}$.

Suppose triangle algorithm for CHMP gives a witness $p' = \begin{pmatrix} x \\ \alpha \end{pmatrix}$. Then,

$$A(-x/\alpha) < b.$$

In other words, triangle algorithm gives complete answer when testing the feasibility of $Ax < b$, not just a yes answer.

Nonstandard Application of Triangle Algorithm: Solving A Linear System

Nonstandard Application of Triangle Algorithm: Solving A Linear System

Consider solving $Ax = b$, with A invertible.

Nonstandard Application of Triangle Algorithm: Solving A Linear System

Consider solving $Ax = b$, with A invertible.
Suppose it is known that $x = A^{-1}b \geq 0$.

Nonstandard Application of Triangle Algorithm: Solving A Linear System

Consider solving $Ax = b$, with A invertible.
Suppose it is known that $x = A^{-1}b \geq 0$.
We can apply the Triangle Algorithm to test if

$$0 \in conv([A, -b]).$$

Nonstandard Application of Triangle Algorithm: Solving A Linear System

Consider solving $Ax = b$, with A invertible.
Suppose it is known that $x = A^{-1}b \geq 0$.
We can apply the Triangle Algorithm to test if

$$0 \in conv([A, -b]).$$

The algorithm produces ε-approximate solution

$$\|Ax_\varepsilon - b\| \leq \varepsilon\|b\|.$$

Incremental Triangle Algorithm: solving $Ax = b$

Incremental Triangle Algorithm: solving $Ax = b$

There exists $t_* \geq 0$ such that for any $t \geq t_*$ the solution of $A(x - te) = b$ is nonnegative (e the vector of ones). Thus

Incremental Triangle Algorithm: solving $Ax = b$

There exists $t_* \geq 0$ such that for any $t \geq t_*$ the solution of $A(x - te) = b$ is nonnegative (e the vector of ones). Thus

$$0 \in conv([A, -(b + tu)]), \quad u = Ae.$$

A convex hull problem is inherent to a linear system.

Incremental Triangle Algorithm: solving $Ax = b$

There exists $t_* \geq 0$ such that for any $t \geq t_*$ the solution of $A(x - te) = b$ is nonnegative (e the vector of ones). Thus

$$0 \in conv([A, -(b + tu)]), \quad u = Ae.$$

A convex hull problem is inherent to a linear system.

Incremental Triangle Algorithm: Given ε, and t (initially zero), test if

$$0 \in conv([A, -(b + tu)]).$$

Incremental Triangle Algorithm: solving $Ax = b$

There exists $t_* \geq 0$ such that for any $t \geq t_*$ the solution of $A(x - te) = b$ is nonnegative (e the vector of ones). Thus

$$0 \in conv([A, -(b + tu)]), \quad u = Ae.$$

A convex hull problem is inherent to a linear system.

Incremental Triangle Algorithm: Given ε, and t (initially zero), test if

$$0 \in conv([A, -(b + tu)]).$$

If $x_t = A^{-1}(b + tu) \geq 0$, Triangle Algorithm produces x_ε satisfying

$$\|Ax_\varepsilon - b\| \leq \varepsilon \|b\|.$$

Otherwise, by the *distance duality*, the algorithm computes a *witness* certifying that $x_t \not\geq 0$.

Incremental Triangle Algorithm: solving $Ax = b$

There exists $t_* \geq 0$ such that for any $t \geq t_*$ the solution of $A(x - te) = b$ is nonnegative (e the vector of ones). Thus

$$0 \in conv([A, -(b + tu)]), \quad u = Ae.$$

A convex hull problem is inherent to a linear system.

Incremental Triangle Algorithm: Given ε, and t (initially zero), test if

$$0 \in conv([A, -(b + tu)]).$$

If $x_t = A^{-1}(b + tu) \geq 0$, Triangle Algorithm produces x_ε satisfying

$$\|Ax_\varepsilon - b\| \leq \varepsilon \|b\|.$$

Otherwise, by the *distance duality*, the algorithm computes a *witness* certifying that $x_t \not\geq 0$. Using the witness, we increment t and repeat.

Numerical Experiments for Solving $Ax = b$

Numerical Experiments for Solving $Ax = b$

In several experiments performed by DIMACS REU student, MS students, a Postdoc: generating different systems, including those from finite difference discretization, Incremental Triangle Algorithm has outperformed Jaobi, Gauss-Seidel, SOR, and AOR, taking much fewer iterations than these methods.

Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix A with nonnegative entries, usually huge but sparse.

Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix A with nonnegative entries, usually huge but sparse.

Usually solved as an eigenvalue problem via the power method.

Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix A with nonnegative entries, usually huge but sparse.

Usually solved as an eigenvalue problem via the power method.

Triangle Algorithm required fewer iterations than the power method.

Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix A with nonnegative entries, usually huge but sparse.

Usually solved as an eigenvalue problem via the power method.

Triangle Algorithm required fewer iterations than the power method.

In some examples triangle algorithm used only one iteration to compute solutions to absolute accuracy 10^{-10}.

Nonstandard Application of Triangle Algorithm: Solving Google PageRank Matrix

The problem is solving $Ax = x$, where $x \geq 0$, $e^T x = 1$, for some square matrix A with nonnegative entries, usually huge but sparse.

Usually solved as an eigenvalue problem via the power method.

Triangle Algorithm required fewer iterations than the power method.

In some examples triangle algorithm used only one iteration to compute solutions to absolute accuracy 10^{-10}. In particular, in an example (from Stanford) where the dimension of A was approximately $300,000$. (Rutgers MS thesis of Hao Shen (2014-2015) includes details.)

Separation of Convex Sets

Separation of Convex Sets

Definition

Given two compact convex subsets K, K' of $\mathbb{R}^m$, we say $H = \{x : h^T x = a\}$ is a separating hyperplane if

$$h^T x < a, \quad \forall x \in K, \quad h^T x < a, \quad \forall x \in K'.$$

Separation of Convex Sets

Definition

Given two compact convex subsets K, K' of $\mathbb{R}^m$, we say $H = \{x : h^T x = a\}$ is a separating hyperplane if

$$h^T x < a, \quad \forall x \in K, \quad h^T x < a, \quad \forall x \in K'.$$

Definition

$$\delta_* = d(K, K') = \min\{d(p, p') : p \in K, p' \in K'\} = d(p_*, p'_*).$$

Separation of Convex Sets

Definition

Given two compact convex subsets K, K' of $\mathbb{R}^m$, we say $H = \{x : h^T x = a\}$ is a separating hyperplane if

$$h^T x < a, \quad \forall x \in K, \quad h^T x < a, \quad \forall x \in K'.$$

Definition

$$\delta_* = d(K, K') = \min\{d(p, p') : p \in K, p' \in K'\} = d(p_*, p'_*).$$

Fact

Then $\delta_* = 0$ if and only if $K \cap K' \neq \emptyset$.

Four Problems Associated to A Pair of Convex Sets

Four Problems Associated to A Pair of Convex Sets

(1) Test if K and K' intersect:

Four Problems Associated to A Pair of Convex Sets

(1) Test if K and K' intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.

Four Problems Associated to A Pair of Convex Sets

(1) Test if K and K' intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.
If K and K' do not intersect:

Four Problems Associated to A Pair of Convex Sets

(1) Test if K and K' intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.
If K and K' do not intersect:
(2) Find a separating hyperplane

Four Problems Associated to A Pair of Convex Sets

(1) Test if K and K' intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.
If K and K' do not intersect:
(2) Find a separating hyperplane
(3) Estimate $\delta_* = d(K, K')$.

Four Problems Associated to A Pair of Convex Sets

(1) Test if K and K' intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.
If K and K' do not intersect:
(2) Find a separating hyperplane
(3) Estimate $\delta_* = d(K, K')$.
(4) Find near-optimal pair of parallel supporting hyperplanes.

Four Problems Associated to A Pair of Convex Sets

(1) Test if K and K' intersect: Find $(p, p') \in K \times K'$ with $d(p, p')$ small.
If K and K' do not intersect:
(2) Find a separating hyperplane
(3) Estimate $\delta_* = d(K, K')$.
(4) Find near-optimal pair of parallel supporting hyperplanes.

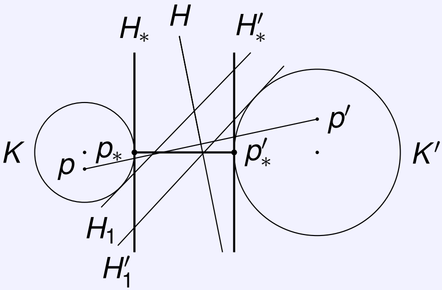


Figure: (p_*, p'_*) optimal pair, (H_*, H'_*) optimal support; (p, p') a pair whose orthogonal bisector separator H; (H_1, H'_1) a supporting pair.

Computing Approximate Intersection Point

Computing Approximate Intersection Point

Definition

Suppose $\delta_* = 0$.

Computing Approximate Intersection Point

Definition

Suppose $\delta_* = 0$. We say a pair $(p, p') \in K \times K'$ is an ε-approximation solution if

Computing Approximate Intersection Point

Definition

Suppose $\delta_* = 0$. We say a pair $(p, p') \in K \times K'$ is an ε-approximation solution if

$$d(p, p') \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

Computing Approximate Intersection Point

Definition

Suppose $\delta_* = 0$. We say a pair $(p, p') \in K \times K'$ is an ε-approximation solution if

$$d(p, p') \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

or

$$d(p, p') \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.$$

Computing Approximate Intersection Point

Definition

Suppose $\delta_* = 0$. We say a pair $(p, p') \in K \times K'$ is an ε-approximation solution if

$$d(p, p') \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

or

$$d(p, p') \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.$$

Definition

Given $(p, p') \in K \times K'$, we say it is a *witness pair* if the orthogonal bisecting hyperplane of the line segment pp' separates K and K'.

Triangle Algorithm I (Testing if K and K' intersect)

Triangle Algorithm I (Testing if K and K' intersect)

Triangle Algorithm I (Testing if K and K' intersect)

The algorithm computes $(p, p') \in K \times K'$ such that

Triangle Algorithm I (Testing if K and K' intersect)

The algorithm computes $(p, p') \in K \times K'$ such that $d(p, p')$ is within a prescribed precision,

Triangle Algorithm I (Testing if K and K' intersect)

The algorithm computes $(p, p') \in K \times K'$ such that $d(p, p')$ is within a prescribed precision,

or $d(p, p')$ is a witness pair.

Pivot Points

Pivot Points

Definition

Given a pair $(p, p') \in K \times K'$,

Pivot Points

Definition

Given a pair $(p, p') \in K \times K'$,
we say $v \in K$ is a p'-pivot for p if

$$d(p, v) \geq d(p', v).$$

Pivot Points

Definition

Given a pair $(p, p') \in K \times K'$,
we say $v \in K$ is a p'-pivot for p if

$$d(p, v) \geq d(p', v).$$

We say $v' \in K'$ is a p-pivot for p' if

$$d(p', v') \geq d(p, v').$$

Pivot Points

Definition

Given a pair $(p, p') \in K \times K'$,
we say $v \in K$ is a p'-pivot for p if

$$d(p, v) \geq d(p', v).$$

We say $v' \in K'$ is a p-pivot for p' if

$$d(p', v') \geq d(p, v').$$

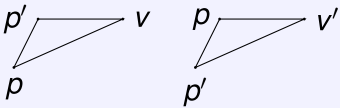


Figure: v is p'-pivot for p (left); v' is p-pivot for p' (right).

Voronoi Diagrams

Voronoi Diagrams

Consider the Voronoi diagram of the two points set $\{p, p'\}$, $(p, p') \in K \times K'$ and the corresponding Voronoi cells:

Voronoi Diagrams

Consider the Voronoi diagram of the two points set $\{p, p'\}$, $(p, p') \in K \times K'$ and the corresponding Voronoi cells:
$V(p) = \{x : d(x, p) < d(x, p')\}, \quad V(p') = \{x : d(x, p') < d(x, p)\}$.

Voronoi Diagrams

Consider the Voronoi diagram of the two points set $\{p, p'\}$, $(p, p') \in K \times K'$ and the corresponding Voronoi cells:
$V(p) = \{x : d(x, p) < d(x, p')\}, \quad V(p') = \{x : d(x, p') < d(x, p)\}$.
Let H be the orthogonal bisecting hyperplane of the line pp'.

Voronoi Diagrams

Consider the Voronoi diagram of the two points set $\{p, p'\}$, $(p, p') \in K \times K'$ and the corresponding Voronoi cells:
$V(p) = \{x : d(x, p) < d(x, p')\}, \quad V(p') = \{x : d(x, p') < d(x, p)\}$.
Let H be the orthogonal bisecting hyperplane of the line pp'.
H intersects $K \iff$ there exists $v \in K$ that is a p'-pivot for p,

Voronoi Diagrams

Consider the Voronoi diagram of the two points set $\{p, p'\}$,
$(p, p') \in K \times K'$ and the corresponding Voronoi cells:
$V(p) = \{x : d(x,p) < d(x,p')\}, \quad V(p') = \{x : d(x,p') < d(x,p)\}.$
Let H be the orthogonal bisecting hyperplane of the line pp'.
H intersects $K \iff$ there exists $v \in K$ that is a p'-pivot for p,
H intersects $K' \iff$ there exists $v' \in K'$ that is a p-pivot for p'.

Voronoi Diagrams

Consider the Voronoi diagram of the two points set $\{p, p'\}$, $(p, p') \in K \times K'$ and the corresponding Voronoi cells:
$V(p) = \{x : d(x, p) < d(x, p')\}, \quad V(p') = \{x : d(x, p') < d(x, p)\}$.
Let H be the orthogonal bisecting hyperplane of the line pp'.
H intersects $K \iff$ there exists $v \in K$ that is a p'-pivot for p,
H intersects $K' \iff$ there exists $v' \in K'$ that is a p-pivot for p'.

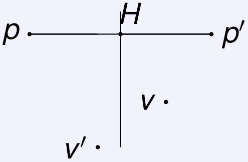


Figure: In the Figure, the point v and v' are pivots for p' and p, respectively.

A New Separating Hyperplane Theorem

A New Separating Hyperplane Theorem

Theorem

(Krein-Milman) *Let K be a compact convex subset of $\mathbb{R}^m$. Then K is the convex hull of its extreme points. In notation, $K = conv(\mathrm{ex}(K))$.*

A New Separating Hyperplane Theorem

A New Separating Hyperplane Theorem

Theorem

(Distance Duality) Let K, K' *be compact convex subsets in* $\mathbb{R}^m$*, with* $\mathrm{ex}(K)$ *and* $\mathrm{ex}(K')$ *as their corresponding set of extreme points. Let* S *be a subset of* K *containing* $\mathrm{ex}(K)$*, and* S' *a subset of* K' *containing* $\mathrm{ex}(K')$*. Then,* $K \cap K' \neq \emptyset$ *if and only if for each* $(p, p') \in K \times K'$*, either there exists* $v \in S$ *such that* $d(p', v) \geq d(p, v)$*, or there exists* $v' \in S'$ *such that* $d(p, v') \geq d(p', v')$*.*

A New Separating Hyperplane Theorem

Theorem

(Distance Duality) Let K, K' be compact convex subsets in $\mathbb{R}^m$, with $\mathrm{ex}(K)$ and $\mathrm{ex}(K')$ as their corresponding set of extreme points. Let S be a subset of K containing $\mathrm{ex}(K)$, and S' a subset of K' containing $\mathrm{ex}(K')$. Then, $K \cap K' \neq \emptyset$ if and only if for each $(p, p') \in K \times K'$, either there exists $v \in S$ such that $d(p', v) \geq d(p, v)$, or there exists $v' \in S'$ such that $d(p, v') \geq d(p', v')$.

An alternative description of the Distance Duality is the following.

Theorem

(Distance Duality) Let K, K' be compact convex subsets in $\mathbb{R}^m$, with $\mathrm{ex}(K)$ and $\mathrm{ex}(K')$ as their corresponding set of extreme points. Then, $K \cap K' = \emptyset$ if and only if there exists $(p, p') \in K \times K'$ such that $d(p, v) < d(p', v)$ for all $v \in \mathrm{ex}(K)$ and $d(p', v') < d(p, v')$ for all $v' \in \mathrm{ex}(K')$. (Such pair is necessarily a witness pair)

Iterative Step in Triangle Algorithm I

Iterative Step in Triangle Algorithm I

Each iteration of Triangle Algorithm I computes for given pair $(p, p') \in K \times K'$, either $v \in K$ that is a p'-pivot for p; or $v' \in K'$, a p-pivot for p'.

Iterative Step in Triangle Algorithm I

Each iteration of Triangle Algorithm I computes for given pair $(p, p') \in K \times K'$, either $v \in K$ that is a p'-pivot for p; or $v' \in K'$, a p-pivot for p'. These are equivalent to checking if

Iterative Step in Triangle Algorithm I

Each iteration of Triangle Algorithm I computes for given pair $(p, p') \in K \times K'$, either $v \in K$ that is a p'-pivot for p; or $v' \in K'$, a p-pivot for p'. These are equivalent to checking if

$$2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.$$

Iterative Step in Triangle Algorithm I

Each iteration of Triangle Algorithm I computes for given pair $(p, p') \in K \times K'$, either $v \in K$ that is a p'-pivot for p; or $v' \in K'$, a p-pivot for p'. These are equivalent to checking if

$$2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.$$

These can be computed by solving the convex programs:

Iterative Step in Triangle Algorithm I

Each iteration of Triangle Algorithm I computes for given pair $(p, p') \in K \times K'$, either $v \in K$ that is a p'-pivot for p; or $v' \in K'$, a p-pivot for p'. These are equivalent to checking if

$$2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.$$

These can be computed by solving the convex programs:

$$\max\{(p' - p)^T v : v \in K\}, \quad \max\{(p - p')^T v' : v' \in K'\}.$$

Iterative Step in Triangle Algorithm I

Each iteration of Triangle Algorithm I computes for given pair $(p, p') \in K \times K'$, either $v \in K$ that is a p'-pivot for p; or $v' \in K'$, a p-pivot for p'. These are equivalent to checking if

$$2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.$$

These can be computed by solving the convex programs:

$$\max\{(p' - p)^T v : v \in K\}, \quad \max\{(p - p')^T v' : v' \in K'\}.$$

Let $T_K, T_{K'}$ be the arithmetic complexities of solving these problems, respectively.

Iterative Step in Triangle Algorithm I

Each iteration of Triangle Algorithm I computes for given pair $(p, p') \in K \times K'$, either $v \in K$ that is a p'-pivot for p; or $v' \in K'$, a p-pivot for p'. These are equivalent to checking if

$$2v^T(p' - p) \geq \|p'\|^2 - \|p\|^2, \quad 2v'^T(p - p') \geq \|p\|^2 - \|p'\|^2.$$

These can be computed by solving the convex programs:

$$\max\{(p' - p)^T v : v \in K\}, \quad \max\{(p - p')^T v' : v' \in K'\}.$$

Let $T_K, T_{K'}$ be the arithmetic complexities of solving these problems, respectively.

Thus the worst-case complexity in each iteration is

$$T = \max\{T_K, T_{K'}\}.$$

Triangle Algorithm I

Triangle Algorithm I ($(p_0, p'_0) \in K \times K'$, $\varepsilon \in (0, 1)$)

- **Step 0.** Set $p = v = p_0$, $p' = v' = p'_0$.
- **Step 1.** If $d(p, p') \le \varepsilon d(p, v)$, or $d(p, p') \le \varepsilon d(p', v')$, stop.
- **Step 2.** Test if there exists $v \in K$ that is a p'-pivot for p, i.e.

  $$2v^T(p' - p) \ge \|p'\|^2 - \|p\|^2$$

  (e.g. by solving $\max\{(p' - p)^T v : v \in K\}$). If such pivot exists, set $p \leftarrow nearest(p'; pv)$, and go to Step 1.
- **Step 3.** Test if there exists $v' \in K'$ that is a p-pivot for p', i.e.

  $$2v'^T(p - p') \ge \|p\|^2 - \|p'\|^2.$$

  (e.g. by solving $\max\{(p - p')^T v' : v' \in K'\}$). If such pivot exists, set $p' \leftarrow nearest(p; p'v')$, and go to Step 1.
- **Step 4.** Output (p, p') as a witness pair, stop ($K \cap K' = \emptyset$).

Triangle Algorithm I

Triangle Algorithm I

When $\delta_* = 0$, the number of iterations to get ε-approximate solution is

$$O\left(\frac{1}{\varepsilon^2}\right).$$

Triangle Algorithm I

When $\delta_* = 0$, the number of iterations to get ε-approximate solution is

$$O\left(\frac{1}{\varepsilon^2}\right).$$

When $\delta_* > 0$, the number of iterations of Triangle Algorithm I to compute a witness pair $(p, p') \in K \times K'$ is

$$O\left(\left(\frac{\rho_*}{\delta_*}\right)^2\right),$$

$$\rho_* = \max\{\Delta_0, \Delta_0'\}, \quad \Delta_0 = \mathrm{diam}(K), \quad \Delta_0' = \mathrm{diam}(K').$$

Testing Separation of Convex Sets

Testing Separation of Convex Sets

Definition

Suppose $\delta_* > 0$.

Testing Separation of Convex Sets

Definition

Suppose $\delta_* > 0$. We say a witness pair $(p, p') \in K \times K'$ is an ε-approximation solution if

$$d(p, p') - \delta_* \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

or

$$d(p, p') - \delta_* \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.$$

Testing Separation of Convex Sets

Definition

Suppose $\delta_* > 0$. We say a witness pair $(p, p') \in K \times K'$ is an ε-approximation solution if

$$d(p, p') - \delta_* \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

or

$$d(p, p') - \delta_* \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.$$

Definition

Suppose $\delta_* > 0$. We say a pair of hyperplanes (H, H') supports (K, K'), if they are parallel, H supports K and and H' supports K'.

Testing Separation of Convex Sets

Testing Separation of Convex Sets

Definition

Suppose $\delta_* > 0$.

Testing Separation of Convex Sets

Definition

Suppose $\delta_* > 0$. We say a witness pair $(p, p') \in K \times K'$ gives an ε-approximate supporting hyperplane, if it is an ε-approximate solution and there exists a pair or supporting hyperplane (H, H'), parallel to the orthogonal bisecting hyperplane of (p, p'), satisfying

$$\delta_* - d(H, H') \leq \varepsilon d(p, v), \quad \text{for some} \quad v \in K,$$

or

$$\delta_* - d(H, H') \leq \varepsilon d(p', v'), \quad \text{for some} \quad v' \in K'.$$

Triangle Algorithm II (Start With a Witness Pair)

Triangle Algorithm II (Start With a Witness Pair)

Given a witness pair $(p, p') \in K \times K'$, it computes an ε-approximate solution, i.e. such that $d(p, p')$ approximates $\delta_* = d(K, K')$.

Triangle Algorithm II (Start With a Witness Pair)

Given a witness pair $(p, p') \in K \times K'$, it computes an ε-approximate solution, i.e. such that $d(p, p')$ approximates $\delta_* = d(K, K')$.

Since (p, p') is a witness-pair, there is no pivot for p, or a pivot for p'.

Triangle Algorithm II (Start With a Witness Pair)

Given a witness pair $(p, p') \in K \times K'$, it computes an ε-approximate solution, i.e. such that $d(p, p')$ approximates $\delta_* = d(K, K')$.

Since (p, p') is a witness-pair, there is no pivot for p, or a pivot for p'.

However, if $d(p, p')$ does not sufficiently approximate δ_*, we will make use of *weak-pivot*, to defined.

Algorithm for Approximation of Distance

Algorithm for Approximation of Distance

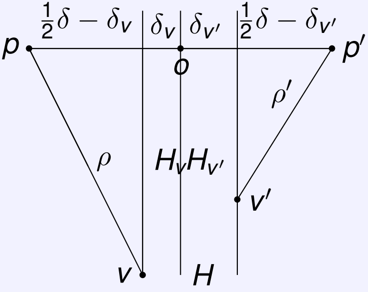


Figure: Depiction of the orthogonal bisector hyperplane H to pp' and parallel supporting hyperplanes H_v and $H_{v'}$ that separate K and K'.

$$\delta_v + \delta_{v'} = d(H_v, H_{v'}) < \delta_* < d(p, p').$$

Algorithm for Approximation of Distance

Algorithm for Approximation of Distance

Theorem

Suppose $(p, p') \in K \times K'$ *is a witness pair. Let the orthogonal bisecting hyperplane to the line* pp' *be* $H = \{x : h^T x = (p - p')^T x = a\}$*. Let*
$v = \mathrm{argmin}\{h^T x : x \in K\}, \quad v' = \mathrm{argmax}\{h^T x : x \in K'\},$

$H_v = \{x : h^T x = h^T v\}, \quad H_{v'} = \{x : h^T x = h^T v'\}.$

Then H_v *and* $H_{v'}$ *are supporting hyperplane to* K *and* K'*, respectively.*

Also, if $\delta_v = d(v, H), \quad \delta_{v'} = d(v', H), \quad \underline{\delta} = \delta_v + \delta_{v'}$*, we have*

$$d(H_v, H_{v'}) = \underline{\delta} = \frac{h^T v - h^T v'}{\|h\|},$$

$$\underline{\delta} \leq \delta_* \leq \delta = d(p, p').$$

Triangle Algorithm II

Triangle Algorithm II

Definition

Given a witness pair $(p, p') \in K \times K'$, let H be the orthogonal bisecting hyperplane of pp'. We shall say $v \in K$ is a *weak p'-pivot* for p if

$$d(p, H) > d(v, H).$$

Similarly, we shall say $v' \in K'$ is a *weak p-pivot* for p' if

$$d(p', H) > d(v', H).$$

Triangle Algorithm II

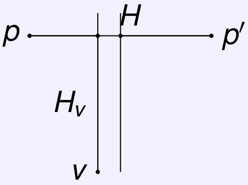
Definition

Given a witness pair $(p, p') \in K \times K'$, let H be the orthogonal bisecting hyperplane of pp'. We shall say $v \in K$ is a *weak p'-pivot* for p if

$$d(p, H) > d(v, H).$$

Similarly, we shall say $v' \in K'$ is a *weak p-pivot* for p' if

$$d(p', H) > d(v', H).$$



Triangle Algorithm II

Triangle Algorithm II

Theorem

Let

$$\Delta_0 = \mathrm{diam}(K), \quad \Delta'_0 = \mathrm{diam}(K'),$$

$$\rho_* = \max\{\Delta_0, \Delta'_0\}.$$

The total arithmetic complexity of Triangle Algorithm II is

$$O\left(T\left(\frac{\rho_*}{\delta_* \varepsilon}\right)^2 \ln \frac{\rho_*}{\delta_*}\right).$$

In particular, when K or K' is a singleton we have

$$O\left(T\left(\frac{\rho_*}{\delta_* \varepsilon}\right)^2\right).$$

Triangle Algorithm II

Triangle Algorithm II

Triangle Algorithm II begins with a witness pair (p_0, p'_0). However, in subsequent iterations the pair $(p_k, p'_k) \in K \times K'$ may or many not be a witness pair.

Triangle Algorithm II

Triangle Algorithm II begins with a witness pair (p_0, p'_0). However, in subsequent iterations the pair $(p_k, p'_k) \in K \times K'$ may or many not be a witness pair.
Thus, the algorithm requires searching for a weak-pivot or a pivot to reduce the gap $\delta_k = d(p_k, p'_k)$ until the desired approximation is attained.

Summary of Triangle Algorithms I and II

Summary of Triangle Algorithms I and II

Let T be the worst-case complexity of computing a pivot for a point in K, or K'. The total number of arithmetic operations in Triangle Algorithm I to get an ε-approximate solution when $\delta_* = 0$, or a witness pair is

$$O\Big(T\frac{1}{\varepsilon^2}\Big).$$

Summary of Triangle Algorithms I and II

Let T be the worst-case complexity of computing a pivot for a point in K, or K'. The total number of arithmetic operations in Triangle Algorithm I to get an ε-approximate solution when $\delta_* = 0$, or a witness pair is

$$O\left(T \frac{1}{\varepsilon^2}\right).$$

The total number of arithmetic operations in Triangle Algorithm II to get an ε-approximate solution to δ_* is

Summary of Triangle Algorithms I and II

Let T be the worst-case complexity of computing a pivot for a point in K, or K'. The total number of arithmetic operations in Triangle Algorithm I to get an ε-approximate solution when $\delta_* = 0$, or a witness pair is

$$O\left(T\frac{1}{\varepsilon^2}\right).$$

The total number of arithmetic operations in Triangle Algorithm II to get an ε-approximate solution to δ_* is

$$O\left(T\left(\frac{\rho_*}{\delta_*}\right)^2\right).$$

And to get and ε-approximate supporting hyperplane is

Summary of Triangle Algorithms I and II

Let T be the worst-case complexity of computing a pivot for a point in K, or K'. The total number of arithmetic operations in Triangle Algorithm I to get an ε-approximate solution when $\delta_* = 0$, or a witness pair is

$$O\left(T\frac{1}{\varepsilon^2}\right).$$

The total number of arithmetic operations in Triangle Algorithm II to get an ε-approximate solution to δ_* is

$$O\left(T\left(\frac{\rho_*}{\delta_*}\right)^2\right).$$

And to get and ε-approximate supporting hyperplane is

$$O\left(T\left(\frac{\rho_*}{\delta_*}\frac{1}{\varepsilon}\right)^2 \ln\frac{\rho_*}{\delta_*}\right).$$

Special Applications and Extensions

Special Applications and Extensions

- When $K = conv(V)$, $V = \{v_1, \ldots, v_n\}$, $K' = conv(V')$, $V' = \{v'_1, \ldots, v'_{n'}\}$). In particular, when one set is a single point. This includes applications such as SVM. In this case

$$T = O(m(n+n')), \quad \text{with preprocessing} \quad T = O(m+\max\{n,n'\}).$$

CS Masters Thesis, Mayank Gupta, 2015-1016, extensive computation and comparison with *sequential minimal optimization* (SMO). The results are very good! Article to be released in near future.

Special Applications and Extensions

- When $K = conv(V)$, $V = \{v_1, \ldots, v_n\}$, $K' = conv(V')$, $V' = \{v'_1, \ldots, v'_{n'}\}$). In particular, when one set is a single point. This includes applications such as SVM. In this case

$$T = O(m(n + n')), \quad \text{with preprocessing} \quad T = O(m + \max\{n, n'\}).$$

  CS Masters Thesis, Mayank Gupta, 2015-1016, extensive computation and comparison with *sequential minimal optimization* (SMO). The results are very good! Article to be released in near future.

- Applications in non-convex optimization.
- Applications in combinatorial and graph problems.
- Applications in conic programming.

Approximation of An NP-Complete Problem

Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix Q, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix Q, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix Q, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix Q, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Qx = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Qx = 0 : x^T x \leq 1\}$.

Let $K = conv(Z)$.

Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix Q, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Let $K = conv(Z)$.

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.

Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix Q, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Qx = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Qx = 0 : x^T x \leq 1\}$.

Let $K = conv(Z)$.

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.

Using the algorithmic separating hyperplane theorem in the corresponding Triangle Algorithm, we can give a fully polynomial-time approximation scheme to either separate S from $conv(Z)$, hence proving that either $Z \cap S$ is empty,

Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix Q, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Let $K = conv(Z)$.

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.

Using the algorithmic separating hyperplane theorem in the corresponding Triangle Algorithm, we can give a fully polynomial-time approximation scheme to either separate S from $conv(Z)$, hence proving that either $Z \cap S$ is empty,

or to give an approximate point in $conv(Z) \cap S$.

Approximation of An NP-Complete Problem

Decision Problem: Given a symmetric $n \times n$ matrix Q, does there exist $x \in S = \{x : e^T x = 1, x \geq 0\}$ such that $x^T Q x = 0$?

This problem is NP-complete.

Let $Z = \{x : x^T Q x = 0 : x^T x \leq 1\}$.

Let $K = conv(Z)$.

Let $K' = S = \{x : e^T x = 1, x \geq 0\}$.

Using the algorithmic separating hyperplane theorem in the corresponding Triangle Algorithm, we can give a fully polynomial-time approximation scheme to either separate S from $conv(Z)$, hence proving that either $Z \cap S$ is empty,

or to give an approximate point in $conv(Z) \cap S$.

In particular, in the later case when Z is convex, the algorithm gives an approximate zero of Q in S.

Related Articles and Forthcoming Work

- A characterization theorem and an algorithm for a convex hull problem, *Annals of operations Research*, Volume 226, Issue 1, pp 301-349, 2014.
- “Finding a Lost Treasure in Convex Hull of Points From Known Distances", Proceedings of the 24th Canadian Conference on Computational Geometry (2012).
- “Three Convex Hull Theorems On Triangles and Circles,” with Jong Youll Park, Honam Mathematical Journal (Korean Journal, in English), 2014.
- “An Algorithmic Separating Hyperplane Theorem and Its Application,” 2014, arxiv.org/pdf/1412.0356v1.pdf.
- Solving Linear System of Equations Via A Convex Hull Algorithm, http://arxiv.org/abs/1210.7858
- On the Triangle Algorithm for Convex Hull Membership, 23rd Fall Workshop on Computational Geometry, City College of New York, Oct 25, 2013, with Michael Saks (2-page abstract).
- Experiments with the Triangle Algorithm for Linear Systems, 23rd Fall Workshop on Computational Geometry, City College of New York, Oct 25, 2013, with Thomas Gibson (2-page abstract).
- Experimental Study of the Convex Hull Decision Problem via a New Geometric Algorithm, 23rd Fall Workshop on Computational Geometry, City College of New York, Oct 25, 2013, with Meng Li. (2-page abstract).
- “Randomized triangle algorithms for convex hull membership, 2-page Extended Abstract in 24nd Annual Fall Workshop on Computational Geometry, Connecticut, 2014.
- A Geometric Polynomial-Time Algorithm for Bipartite Perfect Matching Problem, forthcoming.
- An Approximation to an NP-Complete Problem via The Triangle Algorithm, forthcoming.

- Finally, there remain many open problems.