Learning in the Context of Set Theoretic Estimation: an Efficient and Unifying Framework for Adaptive Machine Learning and Signal Processing

Sergios Theodoridis[1]

a joint work with
K. Slavakis (Univ. of Peloponnese, Greece), and
I. Yamada (Tokyo Institute of Technology, Japan)

[1]University of Athens, Athens, Greece.

Vienna,
April 11th, 2012

"ΟΥΔΕΙΣ ΑΓΕΩΜΕΤΡΗΤΟΣ ΕΙΣΙΤΩ"

“ΟΥΔΕΙΣ ΑΓΕΩΜΕΤΡΗΤΟΣ ΕΙΣΙΤΩ”

(“Those who do not know geometry are not welcome here”)

Plato’s Academy of Philosophy

Part A

Outline of Part A

- The set theoretic estimation approach and multiple intersecting closed convex sets.
- The fundamental tool of metric projections in Hilbert spaces.
- Online classification and regression.
- The concept of Reproducing Kernel Hilbert Spaces (RKHS) and nonlinear processing.
- Distributive learning in sensor networks.

Problem Definition

Given

- A set of measurements $(\boldsymbol{x}_n, y_n)_{n=1}^{N}$, which are jointly distributed, and

Problem Definition

Given

- A set of measurements $(\boldsymbol{x}_n, y_n)_{n=1}^N$, which are jointly distributed, and
- A parametric set of functions $\mathcal{F} = \left\{ f_\alpha(\boldsymbol{x}) : \alpha \in A \subset \mathbb{R}^k \right\}$.

Problem Definition

Given

- A set of measurements $(\boldsymbol{x}_n, y_n)_{n=1}^N$, which are jointly distributed, and
- A parametric set of functions $\mathcal{F} = \left\{ f_\alpha(\boldsymbol{x}) : \alpha \in A \subset \mathbb{R}^k \right\}$.

Compute an $f(\cdot)$, within $\mathcal{F}$, that best approximates y given the value of $\boldsymbol{x}$:

$$y \approx f(\boldsymbol{x}).$$

Problem Definition

Given

- A set of measurements $(\boldsymbol{x}_n, y_n)_{n=1}^N$, which are jointly distributed, and
- A parametric set of functions $\mathcal{F} = \{f_\alpha(\boldsymbol{x}) : \alpha \in A \subset \mathbb{R}^k\}$.

Compute an $f(\cdot)$, within $\mathcal{F}$, that best approximates y given the value of $\boldsymbol{x}$:

$$y \approx f(\boldsymbol{x}).$$

Special Cases

Smoothing, prediction, curve-fitting, regression, classification, filtering, system identification, and beamforming.

The More Classical Approach

Select a loss function $\mathcal{L}(\cdot,\cdot)$ and estimate $f(\cdot)$ so that

$$f(\cdot) \in \arg\min_{f_\alpha(\cdot):\ \alpha\in A} \sum_{n=1}^{N} \mathcal{L}\big(y_n, f_\alpha(\boldsymbol{x}_n)\big).$$

The More Classical Approach

Select a loss function $\mathcal{L}(\cdot,\cdot)$ and estimate $f(\cdot)$ so that

$$f(\cdot) \in \arg\min_{f_\alpha(\cdot):\ \alpha\in A} \sum_{n=1}^{N} \mathcal{L}\big(y_n, f_\alpha(\boldsymbol{x}_n)\big).$$

Drawbacks

- Most often, in practice, the choice of the cost is dictated not by physical reasoning but by computational tractability.

The More Classical Approach

Select a loss function $\mathcal{L}(\cdot,\cdot)$ and estimate $f(\cdot)$ so that

$$f(\cdot) \in \arg\min_{f_\alpha(\cdot):\ \alpha\in A} \sum_{n=1}^{N} \mathcal{L}\big(y_n, f_\alpha(\boldsymbol{x}_n)\big).$$

Drawbacks

- Most often, in practice, the choice of the cost is dictated not by physical reasoning but by computational tractability.
- The existence of a-priori information in the form of constraints makes the task even more difficult.

The More Classical Approach

Select a loss function $\mathcal{L}(\cdot,\cdot)$ and estimate $f(\cdot)$ so that

$$f(\cdot) \in \arg\min_{f_\alpha(\cdot):\ \alpha \in A} \sum_{n=1}^{N} \mathcal{L}\big(y_n, f_\alpha(\boldsymbol{x}_n)\big).$$

Drawbacks

- Most often, in practice, the choice of the cost is dictated not by physical reasoning but by computational tractability.
- The existence of a-priori information in the form of constraints makes the task even more difficult.
- The optimization task is solved iteratively, and iterations freeze after a finite number of steps. Thus, the obtained solution lies in a neighborhood of the optimal one.

The More Classical Approach

Select a loss function $\mathcal{L}(\cdot,\cdot)$ and estimate $f(\cdot)$ so that

$$f(\cdot) \in \arg\min_{f_\alpha(\cdot):\ \alpha\in A} \sum_{n=1}^{N} \mathcal{L}\big(y_n, f_\alpha(\boldsymbol{x}_n)\big).$$

Drawbacks

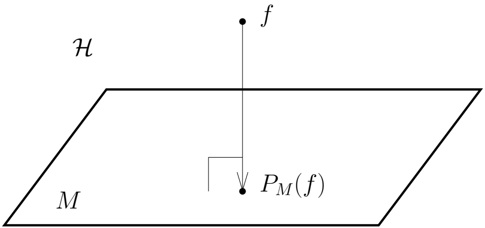
- Most often, in practice, the choice of the cost is dictated not by physical reasoning but by computational tractability.
- The existence of a-priori information in the form of constraints makes the task even more difficult.
- The optimization task is solved iteratively, and iterations freeze after a finite number of steps. Thus, the obtained solution lies in a neighborhood of the optimal one.
- The stochastic nature of the data and the existence of noise add another uncertainty to the optimality of the obtained solution.

- In this talk, we are concerned in finding a set of solutions, which are in agreement with all the available information.
- This will be achieved in the general context of
  - Set theoretic estimation.
  - Convexity.
  - Mappings or operators, e.g., projections, and their associated fixed point sets.

Projection onto a Closed Subspace

Theorem

Given a Euclidean $\mathbb{R}^m$ or a Hilbert space $\mathcal{H}$, the projection of a point f onto a closed subspace M is the unique point $P_M(f) \in M$ that lies closest to f (Pythagoras Theorem).



Projection onto a Closed Convex Set

Theorem

Let C be a closed convex set in a Hilbert space $\mathcal{H}$. Then, for each $f \in \mathcal{H}$, there exists a unique $f_* \in C$ such that

$$\|f - f_*\| = \min_{g \in C} \|f - g\| =: d(f, C).$$

Projection onto a Closed Convex Set

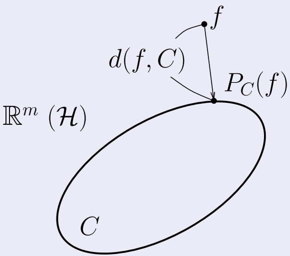
Theorem

Let C be a closed convex set in a Hilbert space $\mathcal{H}$. Then, for each $f \in \mathcal{H}$, there exists a unique $f_* \in C$ such that

$$\|f - f_*\| = \min_{g \in C} \|f - g\| =: d(f, C).$$

Definition (Metric Projection Mapping)

The projection is the mapping $P_C : \mathcal{H} \to C : f \mapsto P_C(f) := f_*$.



Projection onto a Closed Convex Set

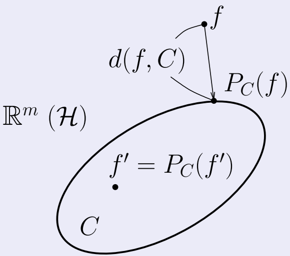
Theorem

Let C be a closed convex set in a Hilbert space $\mathcal{H}$. Then, for each $f \in \mathcal{H}$, there exists a unique $f_* \in C$ such that

$$\|f - f_*\| = \min_{g \in C} \|f - g\| =: d(f, C).$$

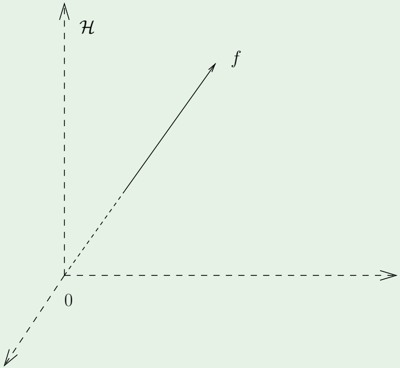
Definition (Metric Projection Mapping)

The projection is the mapping $P_C : \mathcal{H} \to C : f \mapsto P_C(f) := f_*$.

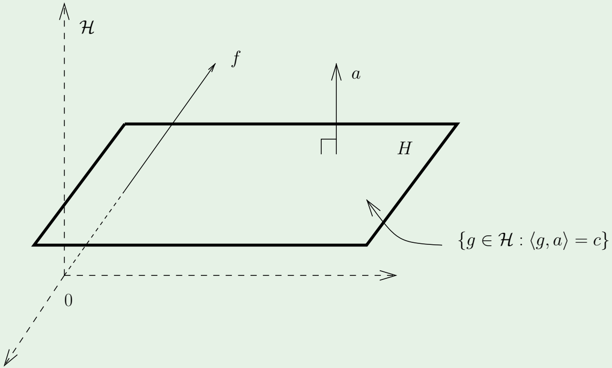


Projection Mappings

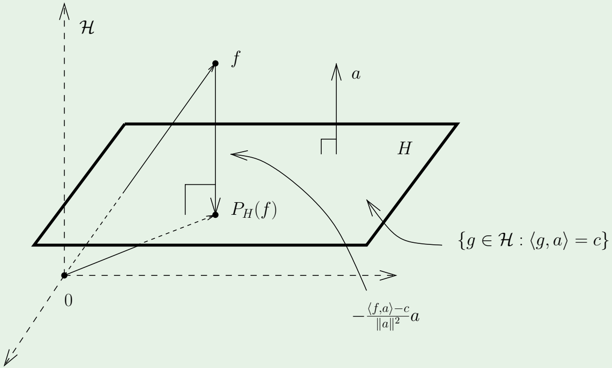
Example (Hyperplane $H := \{g \in \mathcal{H} : \langle g, a \rangle = c\}$)



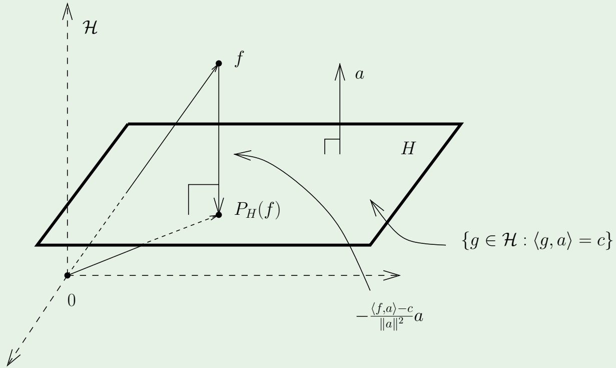
Example (Hyperplane $H := \{g \in \mathcal{H} : \langle g, a \rangle = c\}$)



Example (Hyperplane $H := \{g \in \mathcal{H} : \langle g, a \rangle = c\}$)

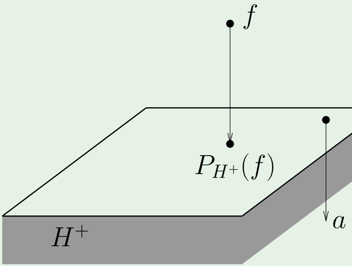


Example (Hyperplane $H := \{g \in \mathcal{H} : \langle g, a\rangle = c\}$)



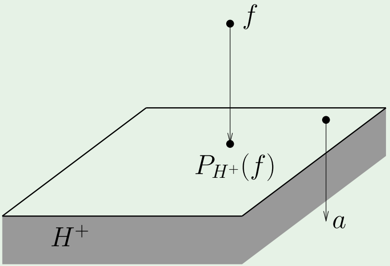
$$P_H(f) = f - \frac{\langle f, a\rangle - c}{\|a\|^2} a, \qquad \forall f \in \mathcal{H}.$$

Example (Halfspace $H^+ := \{g \in \mathcal{H} : \langle g, a \rangle \geq c\}$)



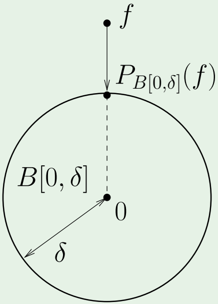
Projection Mappings

Example (Halfspace $H^+ := \{g \in \mathcal{H} : \langle g, a \rangle \geq c\}$)

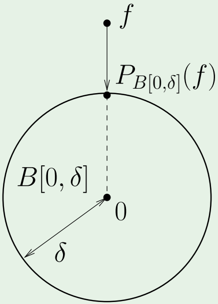


$$P_{H^+}(f) = f - \frac{\min\left\{0, \langle f, a \rangle - c\right\}}{\|a\|^2} a, \qquad \forall f \in \mathcal{H}.$$

Example (Closed Ball $B[0,\delta] := \{g \in \mathcal{H} : \|g\| \leq \delta\}$)

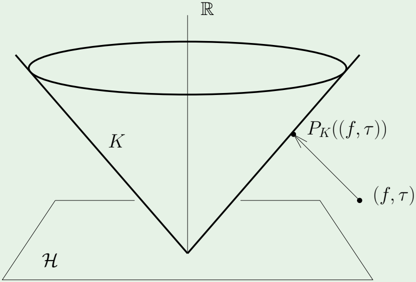


Example (Closed Ball $B[0,\delta] := \{g \in \mathcal{H} : \|g\| \leq \delta\}$)

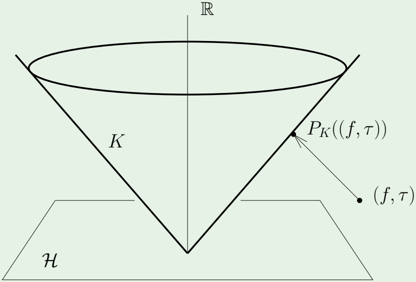


$$P_{B[0,\delta]}(f) := \frac{\delta}{\max\{\delta, \|f\|\}} f, \quad \forall f \in \mathcal{H}.$$

Example (Icecream Cone $K := \left\{(f,\tau) \in \mathcal{H} \times \mathbb{R} : \|f\| \geq \tau\right\}$)



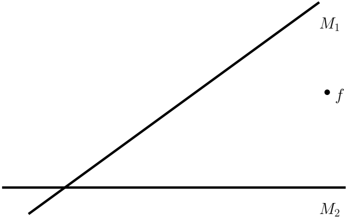
Example (Icecream Cone $K := \{(f,\tau) \in \mathcal{H} \times \mathbb{R} : \|f\| \geq \tau\}$)



$$P_K((f,\tau)) = \begin{cases} (f,\tau), & \text{if } \|f\| \leq \tau, \\ (0,0), & \text{if } \|f\| \leq -\tau, \\ \frac{\|f\|+\tau}{2}\left(\frac{f}{\|f\|}, 1\right), & \text{otherwise}, \end{cases} \qquad \forall (f,\tau) \in \mathcal{H} \times \mathbb{R}.$$

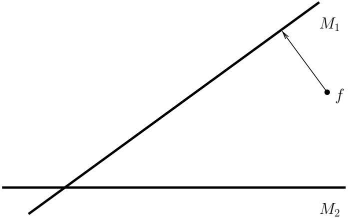
Alternating Projections

Composition of Projection Mappings: Let M_1 and M_2 be closed subspaces in the Hilbert space $\mathcal{H}$. For any $f \in \mathcal{H}$, define the sequence of projections:



Composition of Projection Mappings: Let M_1 and M_2 be closed subspaces in the Hilbert space $\mathcal{H}$. For any $f \in \mathcal{H}$, define the sequence of projections:

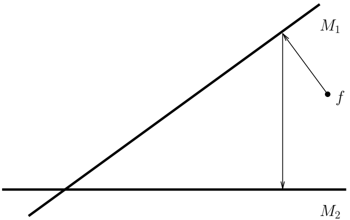
$$P_{M_1}(f).$$



Alternating Projections

Composition of Projection Mappings: Let M_1 and M_2 be closed subspaces in the Hilbert space $\mathcal{H}$. For any $f \in \mathcal{H}$, define the sequence of projections:

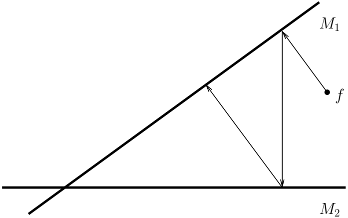
$$P_{M_2} P_{M_1}(f).$$



Alternating Projections

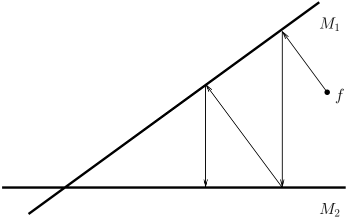
Composition of Projection Mappings: Let M_1 and M_2 be closed subspaces in the Hilbert space $\mathcal{H}$. For any $f \in \mathcal{H}$, define the sequence of projections:

$$P_{M_1} P_{M_2} P_{M_1}(f).$$



Composition of Projection Mappings: Let M_1 and M_2 be closed subspaces in the Hilbert space $\mathcal{H}$. For any $f \in \mathcal{H}$, define the sequence of projections:

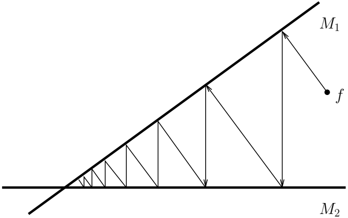
$$P_{M_2} P_{M_1} P_{M_2} P_{M_1}(f).$$



Alternating Projections

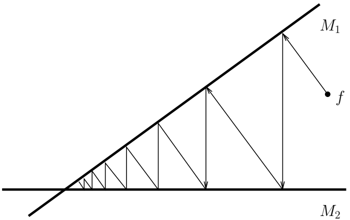
Composition of Projection Mappings: Let M_1 and M_2 be closed subspaces in the Hilbert space $\mathcal{H}$. For any $f \in \mathcal{H}$, define the sequence of projections:

$$\cdots P_{M_2} P_{M_1} P_{M_2} P_{M_1}(f).$$



Composition of Projection Mappings: Let M_1 and M_2 be closed subspaces in the Hilbert space $\mathcal{H}$. For any $f \in \mathcal{H}$, define the sequence of projections:

$$\cdots P_{M_2} P_{M_1} P_{M_2} P_{M_1}(f).$$



Theorem ([von Neumann '33])

For any $f \in \mathcal{H}$, $\lim_{n\to\infty} \left(P_{M_2} P_{M_1}\right)^n (f) = P_{M_1 \cap M_2}(f)$.

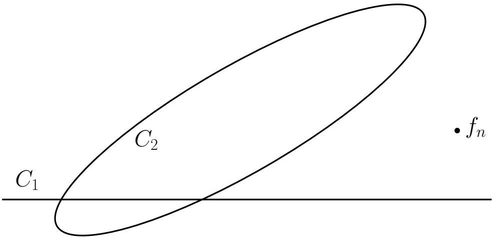
Projections Onto Convex Sets (POCS)

Theorem (POCS[1])

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^p C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. For any $f_0 \in \mathcal{H}$, this defines the sequence of points

$$f_{n+1} := P_{C_p} \cdots P_{C_1}(f_n), \qquad \forall n,$$

converges weakly to an $f_* \in \bigcap_{i=1}^p C_i$.



[1] [Bregman '65], [Gubin, Polyak, Raik '67].

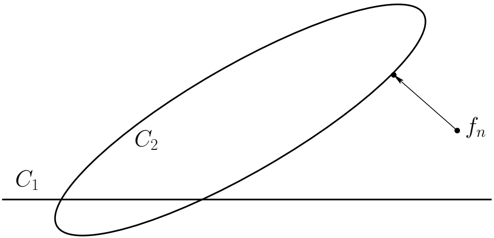
Projections Onto Convex Sets (POCS)

Theorem (POCS[1])

Given a finite number of closed convex sets $C_1, \ldots, C_p$, *with* $\bigcap_{i=1}^{p} C_i \neq \emptyset$, *let their associated projection mappings be* $P_{C_1}, \ldots, P_{C_p}$. *For any* $f_0 \in \mathcal{H}$, *this defines the sequence of points*

$$f_{n+1} := P_{C_p} \cdots P_{C_1}(f_n), \qquad \forall n,$$

converges weakly to an $f_* \in \bigcap_{i=1}^{p} C_i$.



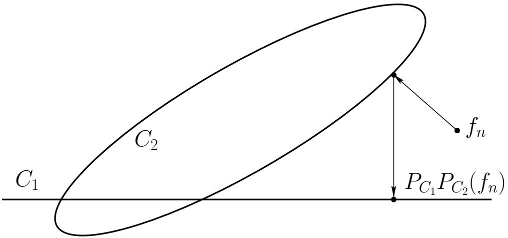
[1] [Bregman '65], [Gubin, Polyak, Raik '67].

Theorem (POCS[1])

Given a finite number of closed convex sets $C_1, \dots, C_p$, *with* $\bigcap_{i=1}^p C_i \neq \emptyset$, *let their associated projection mappings be* $P_{C_1}, \dots, P_{C_p}$. *For any* $f_0 \in \mathcal{H}$, *this defines the sequence of points*

$$f_{n+1} := P_{C_p} \cdots P_{C_1}(f_n), \qquad \forall n,$$

converges weakly to an $f_* \in \bigcap_{i=1}^p C_i$.



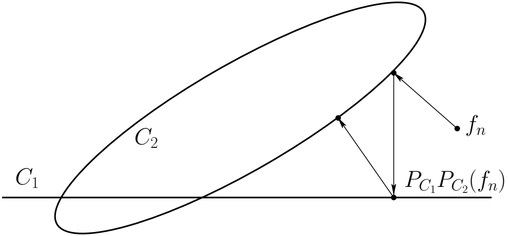
[1] [Bregman '65], [Gubin, Polyak, Raik '67].

Theorem (POCS[1])

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^{p} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. For any $f_0 \in \mathcal{H}$, this defines the sequence of points

$$f_{n+1} := P_{C_p} \cdots P_{C_1}(f_n), \qquad \forall n,$$

converges weakly to an $f_* \in \bigcap_{i=1}^{p} C_i$.



[1] [Bregman '65], [Gubin, Polyak, Raik '67].

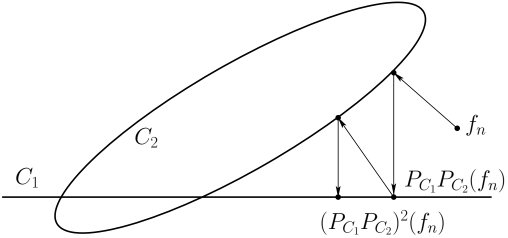
Projections Onto Convex Sets (POCS)

Theorem (POCS[1])

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^{p} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. For any $f_0 \in \mathcal{H}$, this defines the sequence of points

$$f_{n+1} := P_{C_p} \cdots P_{C_1}(f_n), \qquad \forall n,$$

converges weakly to an $f_* \in \bigcap_{i=1}^{p} C_i$.



[1] [Bregman '65], [Gubin, Polyak, Raik '67].

Extrapolated Parallel Projection Method (EPPM)

EPPM[2]

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^{p} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. Let also a set of positive constants $w_1, \ldots, w_p$ such that $\sum_{i=1}^{p} w_i = 1$. Then for any f_0, the sequence

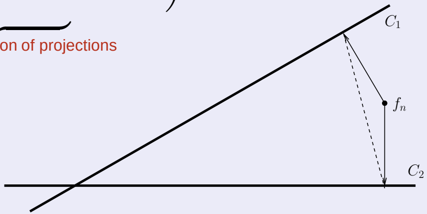
$$f_{n+1} = f_n + \mu_n \Bigg(\underbrace{\sum_{i=1}^{p} w_i P_{C_i}(f_n)}_{\text{Convex combination of projections}} - f_n \Bigg), \quad \forall n,$$

[2][Pierra '84].

EPPM[2]

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^{p} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. Let also a set of positive constants $w_1, \ldots, w_p$ such that $\sum_{i=1}^{p} w_i = 1$. Then for any f_0, the sequence

$$f_{n+1} = f_n + \mu_n \Big(\underbrace{\sum_{i=1}^{p} w_i P_{C_i}(f_n)}_{\text{Convex combination of projections}} - f_n \Big), \quad \forall n,$$

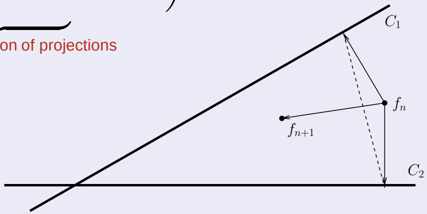


[2] [Pierra '84].

EPPM[2]

Given a finite number of closed convex sets $C_1, \dots, C_p$, with $\bigcap_{i=1}^p C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \dots, P_{C_p}$. Let also a set of positive constants $w_1, \dots, w_p$ such that $\sum_{i=1}^p w_i = 1$. Then for any f_0, the sequence

$$f_{n+1} = f_n + \mu_n \Big(\underbrace{\sum_{i=1}^p w_i P_{C_i}(f_n)}_{\text{Convex combination of projections}} - f_n \Big), \quad \forall n,$$

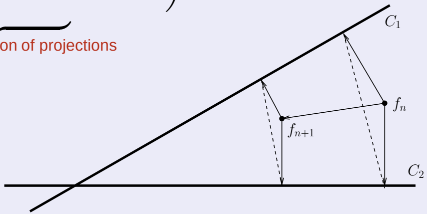


[2] [Pierra '84].

EPPM[2]

Given a finite number of closed convex sets $C_1, \dots, C_p$, with $\bigcap_{i=1}^{p} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \dots, P_{C_p}$. Let also a set of positive constants $w_1, \dots, w_p$ such that $\sum_{i=1}^{p} w_i = 1$. Then for any f_0, the sequence

$$f_{n+1} = f_n + \mu_n \Bigg(\underbrace{\sum_{i=1}^{p} w_i P_{C_i}(f_n)}_{\text{Convex combination of projections}} - f_n \Bigg), \quad \forall n,$$

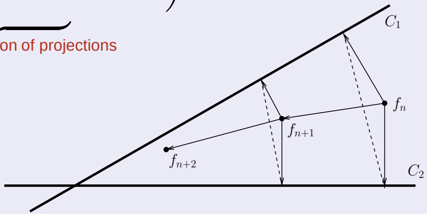


[2] [Pierra '84].

EPPM[2]

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^{p} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. Let also a set of positive constants $w_1, \ldots, w_p$ such that $\sum_{i=1}^{p} w_i = 1$. Then for any f_0, the sequence

$$f_{n+1} = f_n + \mu_n \Bigg(\underbrace{\sum_{i=1}^{p} w_i P_{C_i}(f_n)}_{\text{Convex combination of projections}} - f_n \Bigg), \quad \forall n,$$



[2] [Pierra '84].

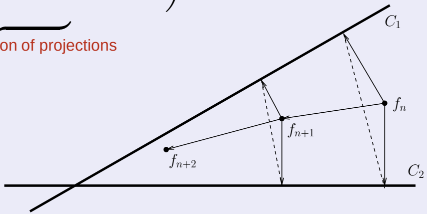
EPPM[2]

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^p C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. Let also a set of positive constants $w_1, \ldots, w_p$ such that $\sum_{i=1}^p w_i = 1$. Then for any f_0, the sequence

$$f_{n+1} = f_n + \mu_n \Bigg(\underbrace{\sum_{i=1}^{p} w_i P_{C_i}(f_n)}_{\text{Convex combination of projections}} - f_n \Bigg), \quad \forall n,$$

converges weakly to a point f_* in $\bigcap_{i=1}^p C_i$, where $\mu_n \in (\epsilon, \mathcal{M}_n)$, for $\epsilon \in (0,1)$, and

$$\mathcal{M}_n := \frac{\sum_{i=1}^p w_i \left\| P_{C_i}(f_n) - f_n \right\|^2}{\left\| \sum_{i=1}^p w_i P_{C_i}(f_n) - f_n \right\|^2}.$$



[2] [Pierra '84].

Infinite Number of Closed Convex Sets

Adaptive Projected Subgradient Method (APSM)[3]

Given an infinite number of closed convex sets $(C_n)_{n\geq 0}$, let their associated projection mappings be $(P_{C_n})_{n\geq 0}$. For any starting point f_0, and an integer $q > 0$, let the sequence

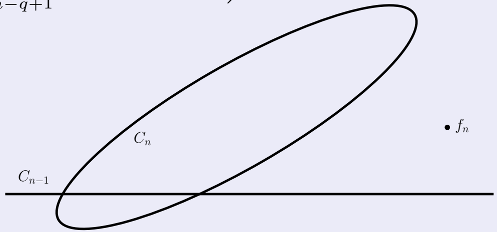
$$f_{n+1} = f_n + \mu_n \left(\sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \right), \quad \forall n,$$

[3] [Yamada '03], [Yamada, Ogura '04], [Slavakis, Yamada, Ogura '06].

Adaptive Projected Subgradient Method (APSM)[3]

Given an infinite number of closed convex sets $(C_n)_{n\geq 0}$, let their associated projection mappings be $(P_{C_n})_{n\geq 0}$. For any starting point f_0, and an integer $q > 0$, let the sequence

$$f_{n+1} = f_n + \mu_n \left(\sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \right), \quad \forall n,$$



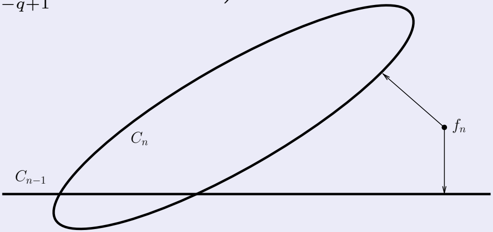
[3] [Yamada '03], [Yamada, Ogura '04], [Slavakis, Yamada, Ogura '06].

Infinite Number of Closed Convex Sets

Adaptive Projected Subgradient Method (APSM)[3]

Given an infinite number of closed convex sets $(C_n)_{n\geq 0}$, let their associated projection mappings be $(P_{C_n})_{n\geq 0}$. For any starting point f_0, and an integer $q > 0$, let the sequence

$$f_{n+1} = f_n + \mu_n \left(\sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \right), \quad \forall n,$$

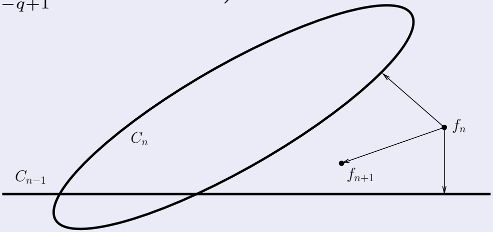


[3] [Yamada '03], [Yamada, Ogura '04], [Slavakis, Yamada, Ogura '06].

Adaptive Projected Subgradient Method (APSM)[3]

Given an infinite number of closed convex sets $(C_n)_{n\geq 0}$, let their associated projection mappings be $(P_{C_n})_{n\geq 0}$. For any starting point f_0, and an integer $q > 0$, let the sequence

$$f_{n+1} = f_n + \mu_n \Bigg(\sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \Bigg), \quad \forall n,$$

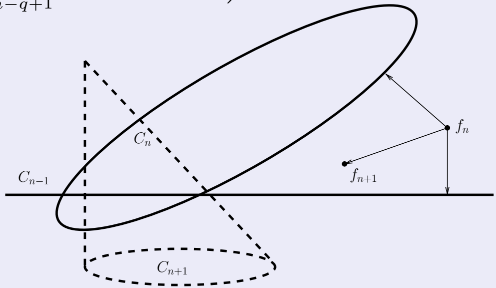


[3] [Yamada '03], [Yamada, Ogura '04], [Slavakis, Yamada, Ogura '06].

Adaptive Projected Subgradient Method (APSM)[3]

Given an infinite number of closed convex sets $(C_n)_{n\geq 0}$, let their associated projection mappings be $(P_{C_n})_{n\geq 0}$. For any starting point f_0, and an integer $q > 0$, let the sequence

$$f_{n+1} = f_n + \mu_n \left(\sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \right), \quad \forall n,$$



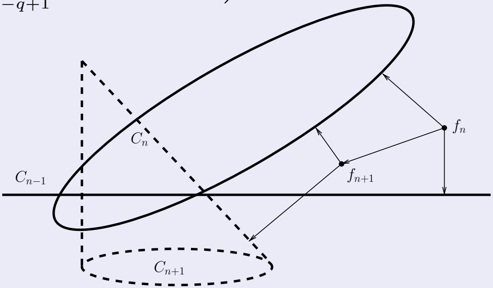
[3] [Yamada '03], [Yamada, Ogura '04], [Slavakis, Yamada, Ogura '06].

Infinite Number of Closed Convex Sets

Adaptive Projected Subgradient Method (APSM)[3]

Given an infinite number of closed convex sets $(C_n)_{n\geq 0}$, let their associated projection mappings be $(P_{C_n})_{n\geq 0}$. For any starting point f_0, and an integer $q > 0$, let the sequence

$$f_{n+1} = f_n + \mu_n \left(\sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \right), \quad \forall n,$$

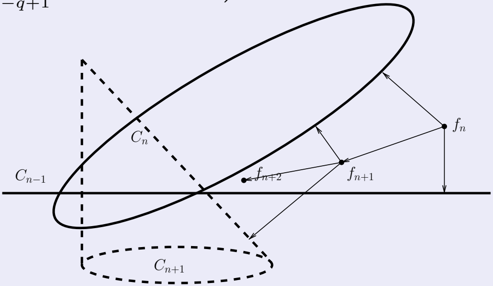


[3] [Yamada '03], [Yamada, Ogura '04], [Slavakis, Yamada, Ogura '06].

Adaptive Projected Subgradient Method (APSM)[3]

Given an infinite number of closed convex sets $(C_n)_{n\geq 0}$, let their associated projection mappings be $(P_{C_n})_{n\geq 0}$. For any starting point f_0, and an integer $q > 0$, let the sequence

$$f_{n+1} = f_n + \mu_n \Bigg(\sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \Bigg), \quad \forall n,$$



[3] [Yamada '03], [Yamada, Ogura '04], [Slavakis, Yamada, Ogura '06].

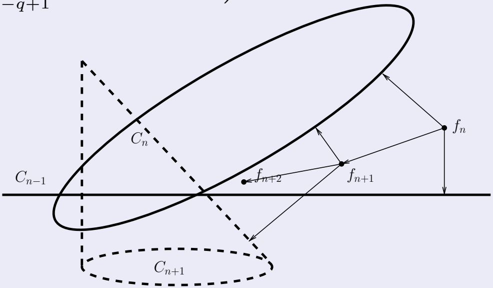
Adaptive Projected Subgradient Method (APSM)[3]

Given an infinite number of closed convex sets $(C_n)_{n\geq 0}$, let their associated projection mappings be $(P_{C_n})_{n\geq 0}$. For any starting point f_0, and an integer $q>0$, let the sequence

$$f_{n+1} = f_n + \mu_n \Big(\sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \Big), \quad \forall n,$$

where $\mu_n \in (0, 2\mathcal{M}_n)$, and $\mathcal{M}_n := \frac{\sum_{j=n-q+1}^{n} w_j \left\| P_{C_j}(f_n) - f_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} w_j P_{C_j}(f_n) - f_n \right\|^2}$.

Under certain constraints the above sequence converges strongly to a point $f_* \in \mathrm{clos}\big(\bigcup_{m\geq 0} \bigcap_{n\geq m} C_n\big)$.



[3] [Yamada '03], [Yamada, Ogura '04], [Slavakis, Yamada, Ogura '06].

The Task

Given a set of training samples $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N \subset \mathbb{R}^m$ and a set of corresponding desired responses $y_0, \ldots, y_N$, estimate a function $f(\cdot) : \mathbb{R}^m \to \mathbb{R}$ that fits the data.

The Task

Given a set of training samples $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N \subset \mathbb{R}^m$ and a set of corresponding desired responses $y_0, \ldots, y_N$, estimate a function $f(\cdot) : \mathbb{R}^m \to \mathbb{R}$ that fits the data.

The Expected / Empirical Risk Function approach

Estimate f so that the expected risk based on a loss function $\mathcal{L}(\cdot, \cdot)$ is minimized:

$$\min_f \mathsf{E}\left\{\mathcal{L}(f(\boldsymbol{x}), y)\right\},$$

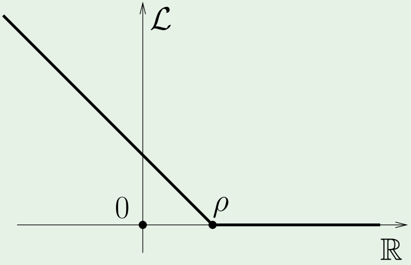
or, in practice, the empirical risk is minimized:

$$\min_f \sum_{n=0}^{N} \mathcal{L}\big(f(\boldsymbol{x}_n), y_n\big).$$

Example (Classification)

For a given margin $\rho \geq 0$, and $y_n \in \{+1, -1\}$, $\forall n$, define the soft margin loss function:

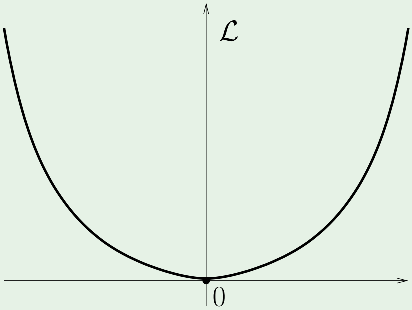
$$\mathcal{L}\big(f(\boldsymbol{x}_n), y_n\big) := \max\big\{0, \rho - y_n f(\boldsymbol{x}_n)\big\}, \qquad \forall n.$$



Example (Regression)

The square loss function:

$$\mathcal{L}\big(f(\boldsymbol{x}_n), y_n\big) := \big(y_n - f(\boldsymbol{x}_n)\big)^2, \qquad \forall n.$$



The Set Theoretic Estimation Approach

Main Idea

The goal here is to have a solution that is in agreement with all the available information, that resides in the data as well as in the available a-priori information.

The Set Theoretic Estimation Approach

Main Idea

The goal here is to have a solution that is in agreement with all the available information, that resides in the data as well as in the available a-priori information.

The Means

- Each piece of information, associated with the training pair $(\boldsymbol{x}_n, y_n)$, is represented in the solution space by a set.

Main Idea

The goal here is to have a solution that is in agreement with all the available information, that resides in the data as well as in the available a-priori information.

The Means

- Each piece of information, associated with the training pair $(\boldsymbol{x}_n, y_n)$, is represented in the solution space by a set.
- Each piece of a-priori information, i.e., each constraint, is also represented by a set.

The Set Theoretic Estimation Approach

Main Idea

The goal here is to have a solution that is in agreement with all the available information, that resides in the data as well as in the available a-priori information.

The Means

- Each piece of information, associated with the training pair $(\boldsymbol{x}_n, y_n)$, is represented in the solution space by a set.
- Each piece of a-priori information, i.e., each constraint, is also represented by a set.
- The intersection of all these sets constitutes the family of solutions.

The Set Theoretic Estimation Approach

Main Idea

The goal here is to have a solution that is in agreement with all the available information, that resides in the data as well as in the available a-priori information.

The Means

- Each piece of information, associated with the training pair $(\boldsymbol{x}_n, y_n)$, is represented in the solution space by a set.
- Each piece of a-priori information, i.e., each constraint, is also represented by a set.
- The intersection of all these sets constitutes the family of solutions.
- The family of solutions is known as the feasibility set.

That is, represent each cost and constraint by an equivalent set C_n and find the solution

$$f \in \bigcap_n C_n \subset \mathcal{H}.$$

Classification: The Soft Margin Loss

The Setting

Let the training data set $(\boldsymbol{x}_n, y_n) \subset \mathbb{R}^m \times \{+1, -1\}$, $n = 0, 1, \ldots$.
Assume the two class task,

$$\begin{cases} y_n = +1, & \boldsymbol{x}_n \in W_1, \\ y_n = -1, & \boldsymbol{x}_n \in W_2. \end{cases}$$

Assume linear separable classes.

Classification: The Soft Margin Loss

The Setting

Let the training data set $(\boldsymbol{x}_n, y_n) \subset \mathbb{R}^m \times \{+1, -1\}$, $n = 0, 1, \ldots$.
Assume the two class task,

$$\begin{cases} y_n = +1, & \boldsymbol{x}_n \in W_1, \\ y_n = -1, & \boldsymbol{x}_n \in W_2. \end{cases}$$

Assume linear separable classes.

The Goal

Classification: The Soft Margin Loss

The Setting

Let the training data set $(\boldsymbol{x}_n, y_n) \subset \mathbb{R}^m \times \{+1, -1\}$, $n = 0, 1, \ldots$.
Assume the two class task,

$$\begin{cases} y_n = +1, & \boldsymbol{x}_n \in W_1, \\ y_n = -1, & \boldsymbol{x}_n \in W_2. \end{cases}$$

Assume linear separable classes.

The Goal

$$\text{Find} \quad f(\boldsymbol{x}) = \boldsymbol{\theta}^t \boldsymbol{x} + b, \quad \text{so that}$$

$$\begin{cases} \boldsymbol{\theta}^t \boldsymbol{x}_n + b \geq \rho, & \text{if } y_n = +1, \\ \boldsymbol{\theta}^t \boldsymbol{x}_n + b \leq \rho, & \text{if } y_n = -1. \end{cases}$$

Classification: The Soft Margin Loss

The Setting

Let the training data set $(\boldsymbol{x}_n, y_n) \subset \mathbb{R}^m \times \{+1, -1\}$, $n = 0, 1, \ldots$.
Assume the two class task,

$$\begin{cases} y_n = +1, & \boldsymbol{x}_n \in W_1, \\ y_n = -1, & \boldsymbol{x}_n \in W_2. \end{cases}$$

Assume linear separable classes.

The Goal

$$\text{Find} \quad f(\boldsymbol{x}) = \boldsymbol{\theta}^t \boldsymbol{x} + b, \quad \text{so that}$$

$$\begin{cases} \boldsymbol{\theta}^t \boldsymbol{x}_n + b \geq \rho, & \text{if } y_n = +1, \\ \boldsymbol{\theta}^t \boldsymbol{x}_n + b \leq \rho, & \text{if } y_n = -1. \end{cases} \quad \text{Hereafter, } \left(\boldsymbol{\theta} \leftarrow \begin{bmatrix} \boldsymbol{\theta} \\ b \end{bmatrix}, \quad \boldsymbol{x}_n \leftarrow \begin{bmatrix} \boldsymbol{x}_n \\ 1 \end{bmatrix}\right).$$

The Piece of Information

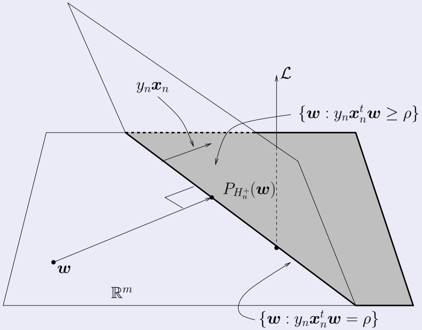
Find all those $\boldsymbol{\theta}$ so that $\quad y_n \boldsymbol{\theta}^t \boldsymbol{x}_n \geq \rho, \quad n = 0, 1, \ldots$

The Piece of Information

Find all those $\boldsymbol{\theta}$ so that $\quad y_n \boldsymbol{\theta}^t \boldsymbol{x}_n \geq \rho, \quad n = 0, 1, \ldots$

The Equivalent Set

$$H_n^+ := \left\{ \boldsymbol{\theta} \in \mathbb{R}^m : y_n \boldsymbol{x}_n^t \boldsymbol{\theta} \geq \rho \right\}, \, n = 0, 1, \ldots.$$



The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.

The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.

The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.
- $P_{H_n^+}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \frac{\min\left\{0, \langle \boldsymbol{\theta}, y_n \boldsymbol{x}_n \rangle\right\}}{\|\boldsymbol{x}_n\|^2} y_n \boldsymbol{x}_n.$

The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.
- $P_{H_n^+}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \frac{\min\left\{0, \langle \boldsymbol{\theta}, y_n \boldsymbol{x}_n \rangle\right\}}{\|\boldsymbol{x}_n\|^2} y_n \boldsymbol{x}_n.$

$\boldsymbol{\theta}_{n-1}$

The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

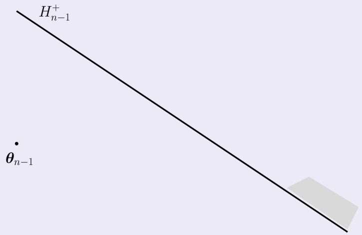
$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.
- $P_{H_n^+}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \frac{\min\left\{0, \langle \boldsymbol{\theta}, y_n \boldsymbol{x}_n \rangle\right\}}{\|\boldsymbol{x}_n\|^2} y_n \boldsymbol{x}_n$.



The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

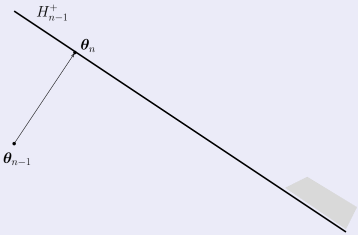
$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.
- $P_{H_n^+}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \frac{\min\left\{0, \langle \boldsymbol{\theta}, y_n \boldsymbol{x}_n \rangle\right\}}{\|\boldsymbol{x}_n\|^2} y_n \boldsymbol{x}_n.$



The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

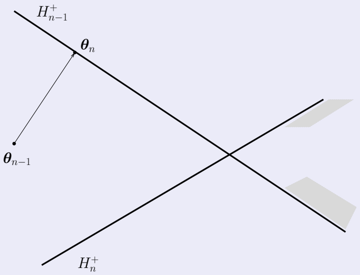
$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.
- $P_{H_n^+}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \frac{\min\left\{0, \langle \boldsymbol{\theta}, y_n \boldsymbol{x}_n \rangle\right\}}{\|\boldsymbol{x}_n\|^2} y_n \boldsymbol{x}_n.$



The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

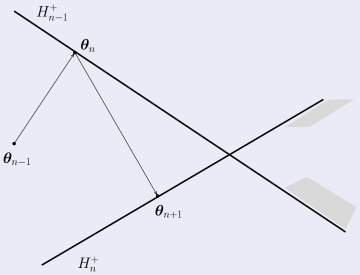
$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.
- $P_{H_n^+}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \frac{\min\left\{0, \langle \boldsymbol{\theta}, y_n \boldsymbol{x}_n \rangle\right\}}{\|\boldsymbol{x}_n\|^2} y_n \boldsymbol{x}_n.$



The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

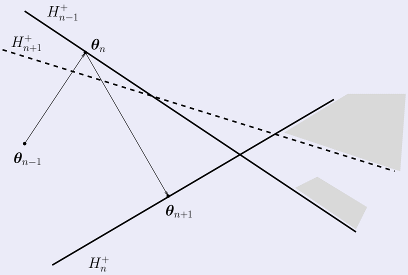
$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.
- $P_{H_n^+}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \frac{\min\{0, \langle \boldsymbol{\theta}, y_n \boldsymbol{x}_n \rangle\}}{\|\boldsymbol{x}_n\|^2} y_n \boldsymbol{x}_n.$



The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent halfspace H_n^+, and

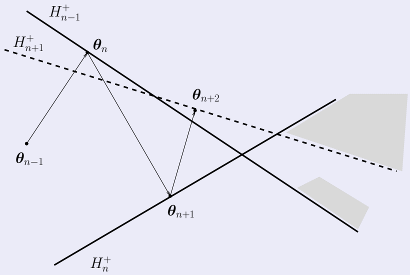
$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n H_n^+.$$

If linearly separable, the problem is feasible.

The Algorithm

Each H_n^+ is a convex set.

- Start from an arbitrary initial $\boldsymbol{\theta}_0$.
- Keep projecting as each H_n^+ is formed.
- $P_{H_n^+}(\boldsymbol{\theta}) = \boldsymbol{\theta} - \frac{\min\left\{0, \langle \boldsymbol{\theta}, y_n \boldsymbol{x}_n \rangle\right\}}{\|\boldsymbol{x}_n\|^2} y_n \boldsymbol{x}_n.$



Algorithmic Solution to Online Classification

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right),$$

$\mu_n \in (0, 2\mathcal{M}_n)$, and

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \left\| P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right),$$

$\mu_n \in (0, 2\mathcal{M}_n)$, and

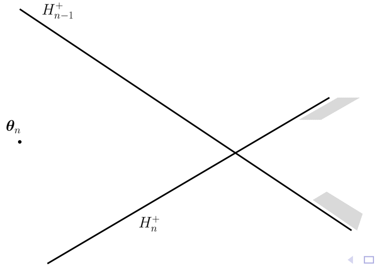
$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \left\| P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

$\boldsymbol{\theta}_n$

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right),$$

$\mu_n \in (0, 2\mathcal{M}_n)$, and

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \left\| P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

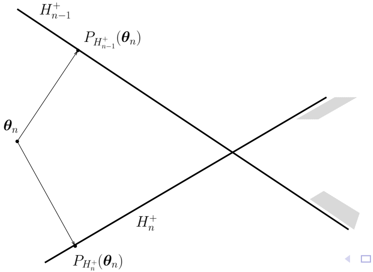


Algorithmic Solution to Online Classification

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right),$$

$\mu_n \in (0, 2\mathcal{M}_n)$, and

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \left\| P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

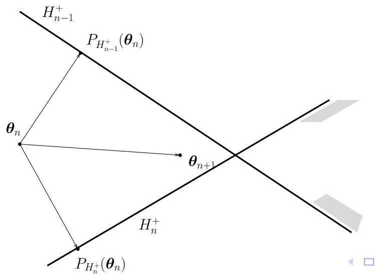


Algorithmic Solution to Online Classification

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right),$$

$\mu_n \in (0, 2\mathcal{M}_n)$, and

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \left\| P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

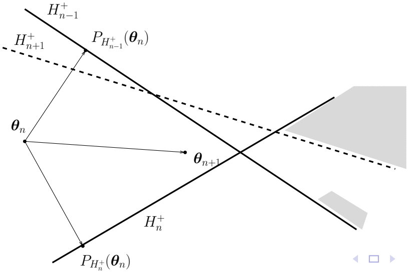


Algorithmic Solution to Online Classification

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right),$$

$\mu_n \in (0, 2\mathcal{M}_n)$, and

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \left\| P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

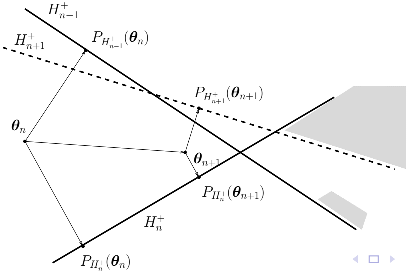


Algorithmic Solution to Online Classification

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right),$$

$\mu_n \in (0, 2\mathcal{M}_n)$, and

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \left\| P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

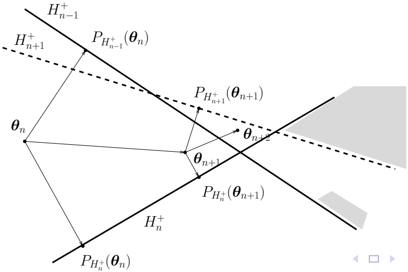


Algorithmic Solution to Online Classification

$$\theta_{n+1} := \theta_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\theta_n) - \theta_n \right),$$

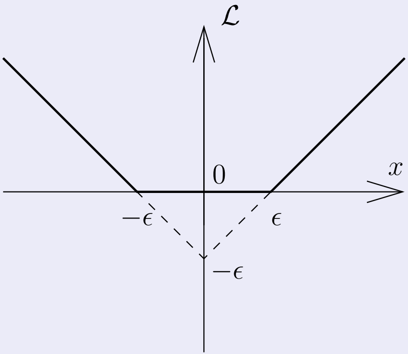
$\mu_n \in (0, 2\mathcal{M}_n)$, and

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \left\| P_{H_n^+}(\theta_n) - \theta_n \right\|^2}{\left\| \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\theta_n) - \theta_n \right\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{H_n^+}(\theta_n) \neq \theta_n, \\ 1, & \text{otherwise.} \end{cases}$$



The linear ϵ-insensitive loss function case

$$\mathcal{L}(x) := \max\{0, |x| - \epsilon\}, \quad x \in \mathbb{R}.$$



The Piece of Information

Given $(\boldsymbol{x}_n, y_n) \in \mathbb{R}^m \times \mathbb{R}$, find $\boldsymbol{\theta} \in \mathbb{R}^m$ such that

$$\left|\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n\right| \leq \epsilon, \qquad \forall n.$$

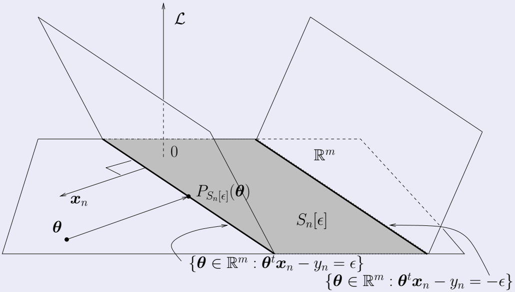
The Piece of Information

Given $(\boldsymbol{x}_n, y_n) \in \mathbb{R}^m \times \mathbb{R}$, find $\boldsymbol{\theta} \in \mathbb{R}^m$ such that

$$\left|\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n\right| \leq \epsilon, \qquad \forall n.$$

The Equivalent Set (Hyperslab)

$$S_n[\epsilon] := \left\{\boldsymbol{\theta} \in \mathbb{R}^m : |\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n| \leq \epsilon\right\}, \quad \forall n.$$



Projection onto a Hyperslab

$$P_{S_n[\epsilon]}(\boldsymbol{\theta}) = \boldsymbol{\theta} + \beta \boldsymbol{x}_n, \quad \forall \boldsymbol{\theta} \in \mathbb{R}^m,$$

where

$$\beta := \begin{cases} \frac{y_n - \boldsymbol{\theta}^t \boldsymbol{x}_n - \epsilon}{\boldsymbol{x}_n^t \boldsymbol{x}_n}, & \text{if } \boldsymbol{\theta}^t \boldsymbol{x}_n - y_n < -\epsilon, \\ 0, & \text{if } \left| \boldsymbol{\theta}^t \boldsymbol{x}_n - y_n \right| \leq \epsilon, \\ -\frac{\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n - \epsilon}{\boldsymbol{x}_n^t \boldsymbol{x}_n}, & \text{if } \boldsymbol{\theta}^t \boldsymbol{x}_n - y_n > \epsilon. \end{cases}$$

Projection onto a Hyperslab

$$P_{S_n[\epsilon]}(\boldsymbol{\theta}) = \boldsymbol{\theta} + \beta \boldsymbol{x}_n, \quad \forall \boldsymbol{\theta} \in \mathbb{R}^m,$$

where

$$\beta := \begin{cases} \frac{y_n - \boldsymbol{\theta}^t \boldsymbol{x}_n - \epsilon}{\boldsymbol{x}_n^t \boldsymbol{x}_n}, & \text{if } \boldsymbol{\theta}^t \boldsymbol{x}_n - y_n < -\epsilon, \\ 0, & \text{if } \left|\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n\right| \leq \epsilon, \\ -\frac{\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n - \epsilon}{\boldsymbol{x}_n^t \boldsymbol{x}_n}, & \text{if } \boldsymbol{\theta}^t \boldsymbol{x}_n - y_n > \epsilon. \end{cases}$$

The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent hyperslab S_n, and

$$\text{find} \quad \boldsymbol{\theta}_* \in \bigcap_n S_n[\epsilon].$$

Algorithm for the Online Regression

Assume weights $\omega_j^{(n)} \geq 0$ such that $\sum_{j=n-q+1}^{n} \omega_j^{(n)} = 1$. For any $\boldsymbol{\theta}_0 \in \mathbb{R}^m$,

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right), \quad \forall n \geq 0,$$

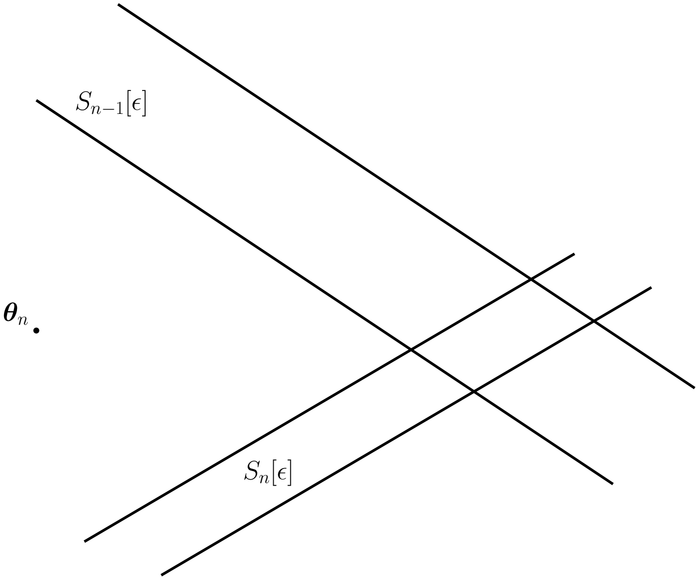
where the extrapolation coefficient $\mu_n \in (0, 2\mathcal{M}_n)$ with

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \|P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\|^2}{\|\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

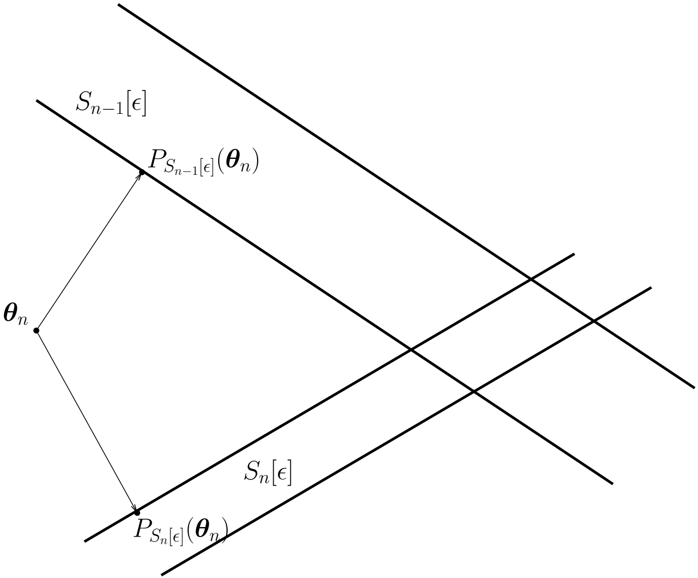
Geometry of the Algorithm

$\boldsymbol{\theta}_n$

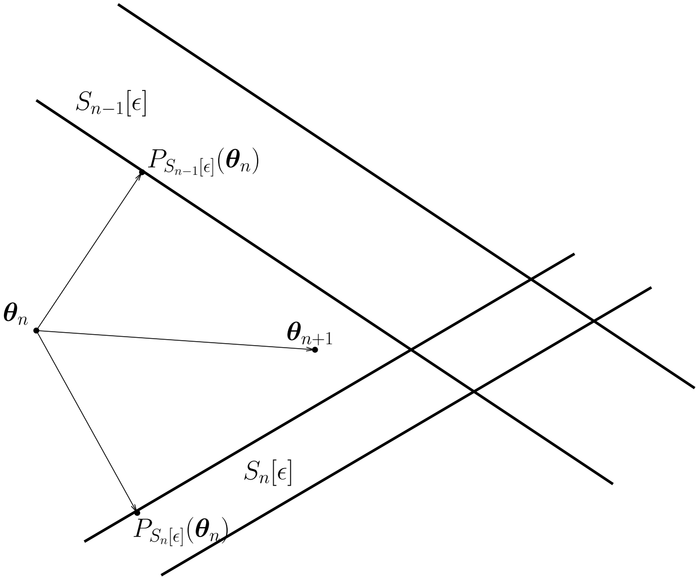
Geometry of the Algorithm



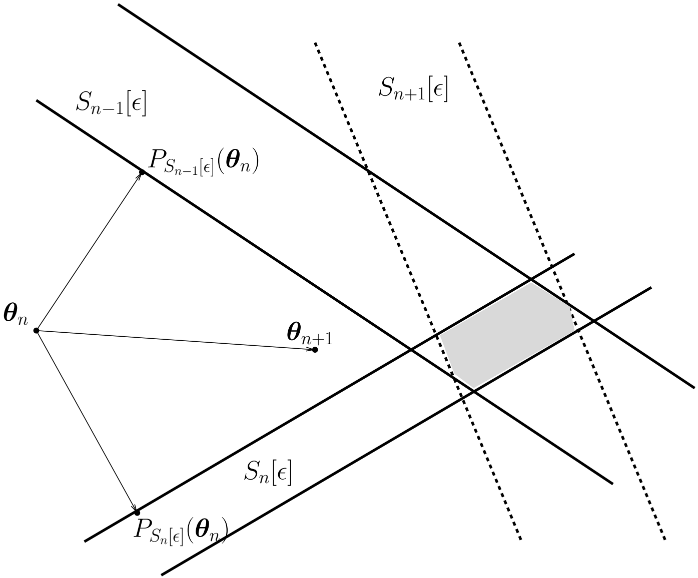
Geometry of the Algorithm

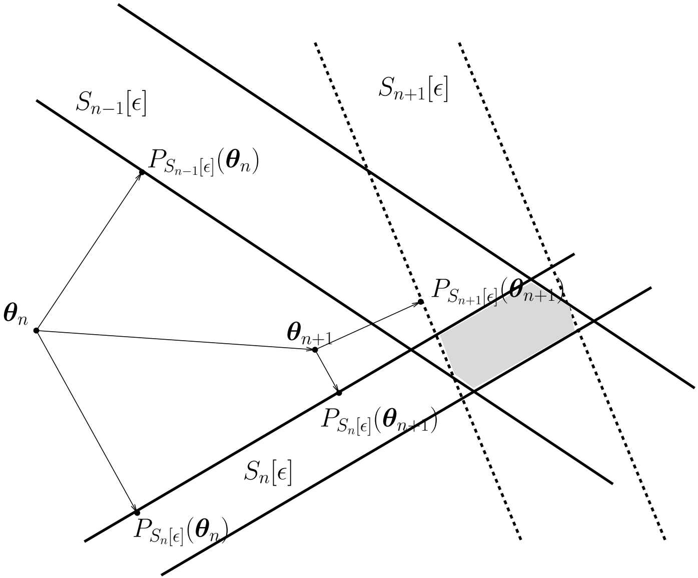


Geometry of the Algorithm



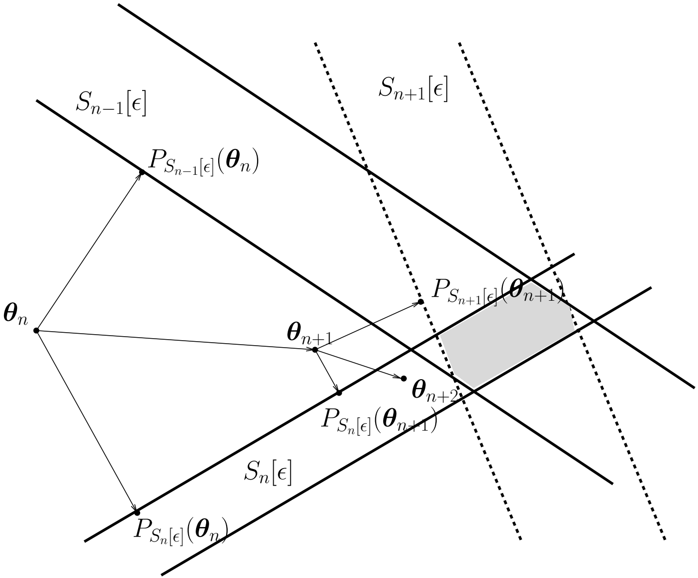
Geometry of the Algorithm





$S_{n-1}[\epsilon]$
$S_{n+1}[\epsilon]$
$P_{S_{n-1}[\epsilon]}(\boldsymbol{\theta}_n)$
$P_{S_{n+1}[\epsilon]}(\boldsymbol{\theta}_{n+1})$
$\boldsymbol{\theta}_n$
$\boldsymbol{\theta}_{n+1}$
$P_{S_n[\epsilon]}(\boldsymbol{\theta}_{n+1})$
$S_n[\epsilon]$
$P_{S_n[\epsilon]}(\boldsymbol{\theta}_n)$

Geometry of the Algorithm



Reproducing Kernel Hilbert Spaces (RKHS)

Definition

Consider a Hilbert space $\mathcal{H}$ of functions $f : \mathbb{R}^m \to \mathbb{R}$.

Reproducing Kernel Hilbert Spaces (RKHS)

Definition

Consider a Hilbert space $\mathcal{H}$ of functions $f : \mathbb{R}^m \to \mathbb{R}$.
Assume there exists a kernel function $\kappa(\cdot, \cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ such that

Reproducing Kernel Hilbert Spaces (RKHS)

Definition

Consider a Hilbert space $\mathcal{H}$ of functions $f : \mathbb{R}^m \to \mathbb{R}$.
Assume there exists a kernel function $\kappa(\cdot, \cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ such that

- $\kappa(\boldsymbol{x}, \cdot) \in \mathcal{H}, \forall \boldsymbol{x} \in \mathbb{R}^m$,

Reproducing Kernel Hilbert Spaces (RKHS)

Definition

Consider a Hilbert space $\mathcal{H}$ of functions $f : \mathbb{R}^m \to \mathbb{R}$.
Assume there exists a kernel function $\kappa(\cdot,\cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ such that

- $\kappa(\boldsymbol{x},\cdot) \in \mathcal{H}$, $\forall \boldsymbol{x} \in \mathbb{R}^m$,
- $\langle f, \kappa(\boldsymbol{x},\cdot)\rangle = f(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathbb{R}^m$, $\forall f \in \mathcal{H}$, (reproducing property).

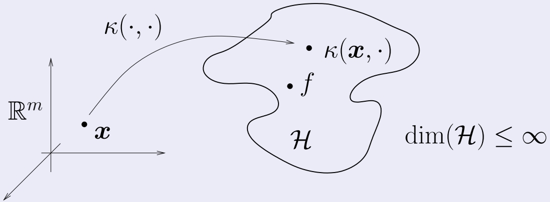
Reproducing Kernel Hilbert Spaces (RKHS)

Definition

Consider a Hilbert space $\mathcal{H}$ of functions $f : \mathbb{R}^m \rightarrow \mathbb{R}$.
Assume there exists a kernel function $\kappa(\cdot,\cdot) : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ such that

- $\kappa(\boldsymbol{x},\cdot) \in \mathcal{H}$, $\forall \boldsymbol{x} \in \mathbb{R}^m$,
- $\langle f, \kappa(\boldsymbol{x},\cdot)\rangle = f(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathbb{R}^m$, $\forall f \in \mathcal{H}$, (reproducing property).

Then $\mathcal{H}$ is called a Reproducing Kernel Hilbert Space (RKHS).



Properties of the Kernel Function

- If such a kernel function exists, then it is a symmetric and positive definite kernel; for any real numbers $a_0, a_1, \ldots, a_N$, any $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots \boldsymbol{x}_N \in \mathbb{R}^m$, and any N,

$$\sum_{i=0}^{N} \sum_{j=0}^{N} a_i a_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.$$

Properties of the Kernel Function

- If such a kernel function exists, then it is a symmetric and positive definite kernel; for any real numbers $a_0, a_1, \ldots, a_N$, any $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots \boldsymbol{x}_N \in \mathbb{R}^m$, and any N,

$$\sum_{i=0}^{N} \sum_{j=0}^{N} a_i a_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.$$

- The reverse is also true. Let

$$\kappa(\cdot, \cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R},$$

be symmetric and positive definite. Then, there exists an RKHS of functions on $\mathbb{R}^m$, such that $\kappa(\cdot, \cdot)$ is a reproducing kernel of $\mathcal{H}$.

Properties of the Kernel Function

- If such a kernel function exists, then it is a symmetric and positive definite kernel; for any real numbers $a_0, a_1, \dots, a_N$, any $\boldsymbol{x}_0, \boldsymbol{x}_1, \dots \boldsymbol{x}_N \in \mathbb{R}^m$, and any N,

$$\sum_{i=0}^{N} \sum_{j=0}^{N} a_i a_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0.$$

- The reverse is also true. Let

$$\kappa(\cdot, \cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R},$$

  be symmetric and positive definite. Then, there exists an RKHS of functions on $\mathbb{R}^m$, such that $\kappa(\cdot, \cdot)$ is a reproducing kernel of $\mathcal{H}$.
- Each RKHS is uniquely defined by a $\kappa(\cdot, \cdot)$, and each (symmetric) positive definite kernel, $\kappa(\cdot, \cdot)$, uniquely defines an RKHS[4].

---

[4] [Aronszajn '50]

- The celebrated kernel trick is formed as follows.

- The celebrated kernel trick is formed as follows. Let

$$\boldsymbol{x} \mapsto \kappa(\boldsymbol{x}, \cdot) =: \phi(\boldsymbol{x}) \in \mathcal{H},$$
$$\boldsymbol{y} \mapsto \kappa(\boldsymbol{y}, \cdot) =: \phi(\boldsymbol{y}) \in \mathcal{H}.$$

- The celebrated kernel trick is formed as follows. Let

$$\boldsymbol{x} \mapsto \kappa(\boldsymbol{x}, \cdot) =: \phi(\boldsymbol{x}) \in \mathcal{H},$$
$$\boldsymbol{y} \mapsto \kappa(\boldsymbol{y}, \cdot) =: \phi(\boldsymbol{y}) \in \mathcal{H}.$$

Then,

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle = \kappa(\boldsymbol{x}, \boldsymbol{y}).$$

- The celebrated kernel trick is formed as follows. Let

$$\boldsymbol{x} \mapsto \kappa(\boldsymbol{x}, \cdot) =: \phi(\boldsymbol{x}) \in \mathcal{H},$$
$$\boldsymbol{y} \mapsto \kappa(\boldsymbol{y}, \cdot) =: \phi(\boldsymbol{y}) \in \mathcal{H}.$$

Then,

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle = \kappa(\boldsymbol{x}, \boldsymbol{y}).$$

- This is an important property since it leads to an easy, black box rule, which transforms a nonlinear task to a linear one; this is done by the following steps...

Steps for Kernel Methods

- Assume the implicit mapping

$$\mathbb{R}^m \ni \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) \in \mathcal{H}.$$

Steps for Kernel Methods

- Assume the implicit mapping

$$\mathbb{R}^m \ni \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) \in \mathcal{H}.$$

- Solve the problem linearly in $\mathcal{H}$.

Steps for Kernel Methods

- Assume the implicit mapping

$$\mathbb{R}^m \ni \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) \in \mathcal{H}.$$

- Solve the problem linearly in $\mathcal{H}$.
- Use an algorithm that can be casted (modified) in terms of inner products.

Steps for Kernel Methods

- Assume the implicit mapping

$$\mathbb{R}^m \ni \boldsymbol{x} \mapsto \phi(\boldsymbol{x}) \in \mathcal{H}.$$

- Solve the problem linearly in $\mathcal{H}$.
- Use an algorithm that can be casted (modified) in terms of inner products.
- Replace inner product computations with kernel ones:

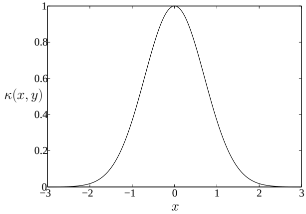
$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle = \kappa(\boldsymbol{x}, \boldsymbol{y}).$$

  This is the step that brings the nonlinearity in the modeling.

Kernel Functions Examples

- The Gaussian kernel:

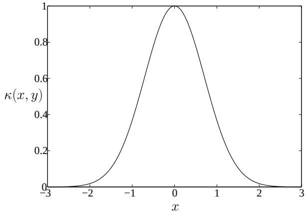
$$\kappa(\boldsymbol{x}, \boldsymbol{y}) := \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma^2}\right),$$



Kernel Functions Examples

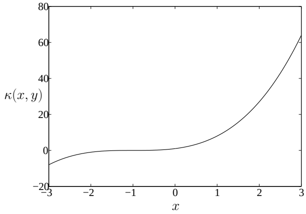
- The Gaussian kernel:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) := \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma^2}\right),$$



- The polynomial kernel:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) := \left(\boldsymbol{x}^t \boldsymbol{y} + 1\right)^d,$$



The Representer Theorem

- Let a strictly monotone increasing function: $\Omega : [0, \infty) \to \mathbb{R}$,

The Representer Theorem

- Let a strictly monotone increasing function: $\Omega : [0, \infty) \to \mathbb{R}$,
- and a (cost) function: $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \cup \{\infty\}$.

The Representer Theorem

- Let a strictly monotone increasing function: $\Omega : [0, \infty) \to \mathbb{R}$,
- and a (cost) function: $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \cup \{\infty\}$.
- Then, the solution of the task

$$\min_{f \in \mathcal{H}} \sum_{n=0}^{N} \mathcal{L}(y_n, f(\boldsymbol{x}_n)) + \Omega(\|f\|),$$

admits a representation of the form:

$$f_* = \sum_{n=0}^{N} a_n \kappa(\boldsymbol{x}_n, \cdot).$$

- Let a strictly monotone increasing function: $\Omega : [0, \infty) \to \mathbb{R}$,
- and a (cost) function: $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R} \cup \{\infty\}$.
- Then, the solution of the task

$$\min_{f \in \mathcal{H}} \sum_{n=0}^{N} \mathcal{L}(y_n, f(\boldsymbol{x}_n)) + \Omega(\|f\|),$$

admits a representation of the form:

$$f_* = \sum_{n=0}^{N} a_n \kappa(\boldsymbol{x}_n, \cdot).$$

Example

$$\mathcal{L}(y_n, f(\boldsymbol{x}_n)) := \big(y_n - f(\boldsymbol{x}_n)\big)^2,$$
$$\Omega(\|f\|) := \|f\|^2 = \langle f, f \rangle.$$

The Goal

Let the training data set $(\boldsymbol{x}_n, y_n) \subset \mathbb{R}^m \times \mathbb{R}$, $n = 0, 1, \ldots$.

- $\boldsymbol{x}_n \mapsto \kappa(\boldsymbol{x}_n, \cdot)$, which is a function of one variable.

Regression in RKHS

The Goal

Let the training data set $(\boldsymbol{x}_n, y_n) \subset \mathbb{R}^m \times \mathbb{R}$, $n = 0, 1, \ldots$.

- $\boldsymbol{x}_n \mapsto \kappa(\boldsymbol{x}_n, \cdot)$, which is a function of one variable.
- Find $f \in \mathcal{H}$ such that

$$|f(\boldsymbol{x}_n) - y_n| \leq \epsilon, \qquad \forall n.$$

Set Theoretic Estimation Approach to Regression

The Piece of Information

Given $(\boldsymbol{x}_n, y_n) \in \mathbb{R}^m \times \mathbb{R}$, $n = 0, 1, 2, \ldots$, find $f \in \mathcal{H}$ such that

$$|\langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n| \leq \epsilon, \qquad \forall n.$$

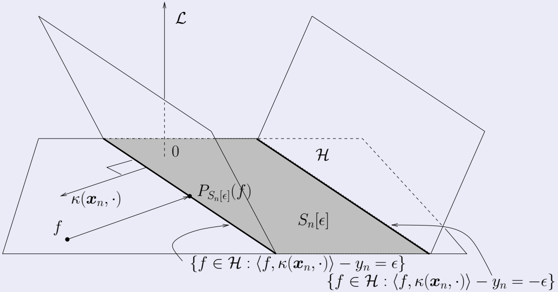
The Piece of Information

Given $(\boldsymbol{x}_n, y_n) \in \mathbb{R}^m \times \mathbb{R}$, $n = 0, 1, 2, \ldots$, find $f \in \mathcal{H}$ such that

$$|\langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n| \leq \epsilon, \qquad \forall n.$$

The Equivalent Set (Hyperslab)

$$S_n[\epsilon] := \left\{f \in \mathcal{H} : |\langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n| \leq \epsilon\right\}, \quad \forall n.$$



Projection onto a Hyperslab

$$P_{S_n[\epsilon]}(f) = f + \beta\kappa(\boldsymbol{x}_n, \cdot), \forall f \in \mathcal{H},$$

where

$$\beta := \begin{cases} \frac{y_n - \langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - \epsilon}{\kappa(\boldsymbol{x}_n, \boldsymbol{x_n})}, & \text{if } \langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n < -\epsilon, \\ 0, & \text{if } |\langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n| \leq \epsilon, \\ -\frac{\langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n - \epsilon}{\kappa(\boldsymbol{x}_n, \boldsymbol{x_n})}, & \text{if } \langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n > \epsilon. \end{cases}$$

Projection onto a Hyperslab

$$P_{S_n[\epsilon]}(f) = f + \beta\kappa(\boldsymbol{x}_n, \cdot), \forall f \in \mathcal{H},$$

where

$$\beta := \begin{cases} \frac{y_n - \langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - \epsilon}{\kappa(\boldsymbol{x}_n, \boldsymbol{x}_n)}, & \text{if } \langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n < -\epsilon, \\ 0, & \text{if } |\langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n| \leq \epsilon, \\ -\frac{\langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n - \epsilon}{\kappa(\boldsymbol{x}_n, \boldsymbol{x}_n)}, & \text{if } \langle f, \kappa(\boldsymbol{x}_n, \cdot)\rangle - y_n > \epsilon. \end{cases}$$

The feasibility set

For each pair $(\boldsymbol{x}_n, y_n)$, form the equivalent hyperslab S_n, and

$$\text{find} \quad f_* \in \bigcap_{n \geq n_0} S_n[\epsilon].$$

Algorithm for Online Regression in RKHS

For $f_0 \in \mathcal{H}$, execute the following algorithm[5]

$$f_{n+1} := f_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(f_n) - f_n \right), \quad \forall n \geq 0,$$

where the extrapolation coefficient $\mu_n \in (0, 2\mathcal{M}_n)$ with

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \|P_{S_j[\epsilon]}(f_n) - f_n\|^2}{\|\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(f_n) - f_n\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(f_n) \neq f_n, \\ 1, & \text{otherwise.} \end{cases}$$

[5] [Slavakis, Theodoridis, Yamada '09].

Sparsification

As time goes by:

$$f_n := \sum_{i=0}^{n-1} \gamma_i^{(n)} \kappa(\boldsymbol{x}_i, \cdot).$$

As time goes by:

$$f_n := \sum_{i=0}^{n-1} \gamma_i^{(n)} \kappa(\boldsymbol{x}_i, \cdot).$$

Memory and computational load grows unbounded as $n \to \infty$!

As time goes by:

$$f_n := \sum_{i=0}^{n-1} \gamma_i^{(n)} \kappa(\boldsymbol{x}_i, \cdot).$$

Memory and computational load grows unbounded as $n \to \infty$!

To cope with the problem, we additionally constrain the norm of f_n by a predefined $\delta > 0$[6]:

$$\forall n \geq 0, \quad f_n \in B[0, \delta] := \{f \in \mathcal{H} : \|f\| \leq \delta\} : \text{ Closed Ball.}$$

[6] [Slavakis, Theodoridis, Yamada '08], [Slavakis, Theodoridis '08].

As time goes by:

$$f_n := \sum_{i=0}^{n-1} \gamma_i^{(n)} \kappa(\boldsymbol{x}_i, \cdot).$$

Memory and computational load grows unbounded as $n \to \infty$!

To cope with the problem, we additionally constrain the norm of f_n by a predefined $\delta > 0$[6]:

$$\forall n \geq 0, \quad f_n \in B[0, \delta] := \{f \in \mathcal{H} : \|f\| \leq \delta\} : \text{ Closed Ball.}$$

Goal

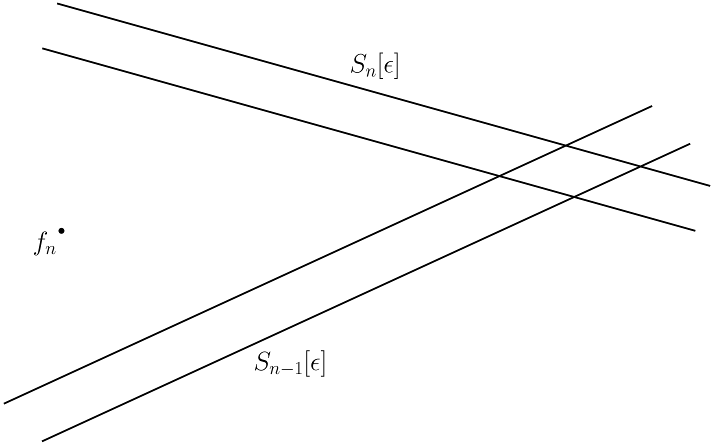
Thus, we are looking for a classifier $f \in \mathcal{H}$ such that

$$f \in B[0, \delta] \cap \Big(\bigcap_{n \geq n_0} S_n[\epsilon] \Big).$$

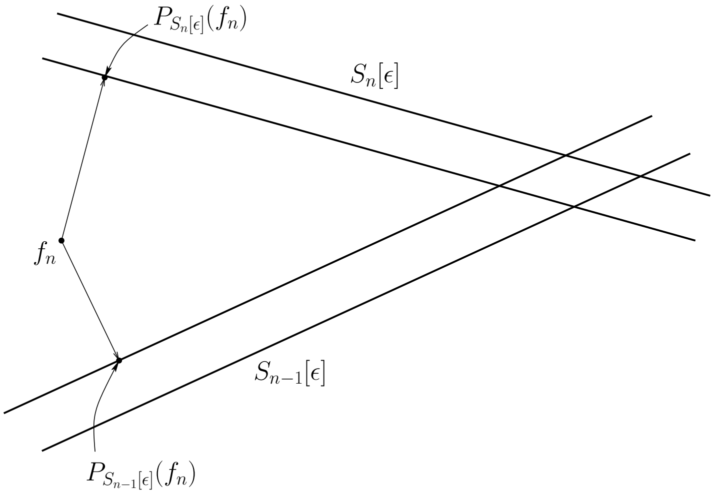
[6] [Slavakis, Theodoridis, Yamada '08], [Slavakis, Theodoridis '08].

Geometric Illustration of the Algorithm

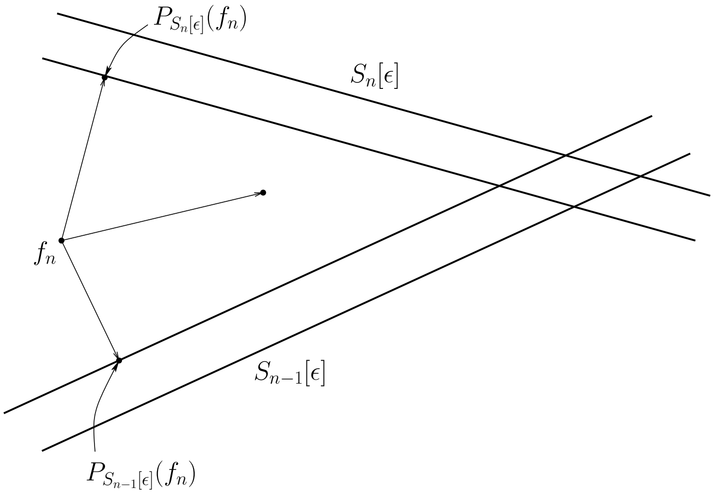
Geometric Illustration of the Algorithm



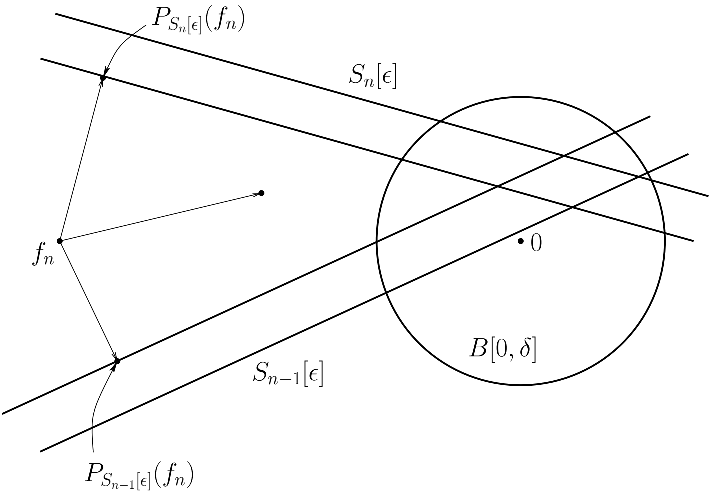
Geometric Illustration of the Algorithm



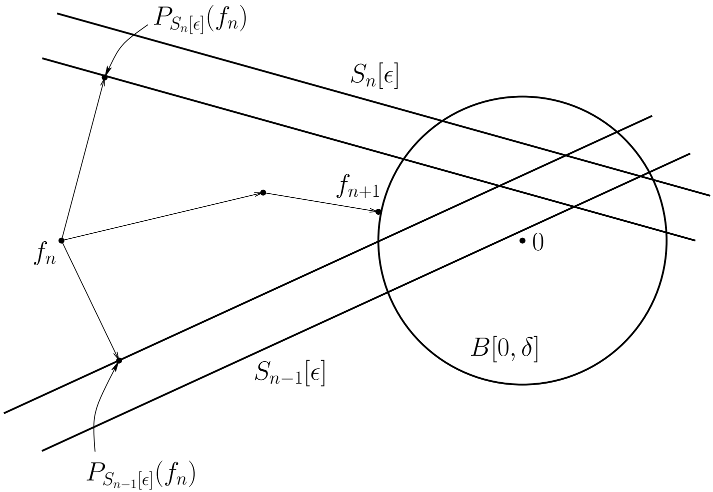
Geometric Illustration of the Algorithm



Geometric Illustration of the Algorithm



Geometric Illustration of the Algorithm



Distributive Learning for Sensor Networks

Problem Definition

- In a distributed network, the nodes are tasked to collect information and estimate a parameter of interest. The general concept can be summarized as follows:

Distributive Learning for Sensor Networks

Problem Definition

- In a distributed network, the nodes are tasked to collect information and estimate a parameter of interest. The general concept can be summarized as follows:
  - The nodes sense an amount of data from the environment.

Problem Definition

- In a distributed network, the nodes are tasked to collect information and estimate a parameter of interest. The general concept can be summarized as follows:
  - The nodes sense an amount of data from the environment.
  - Computations are performed locally in each node.

Problem Definition

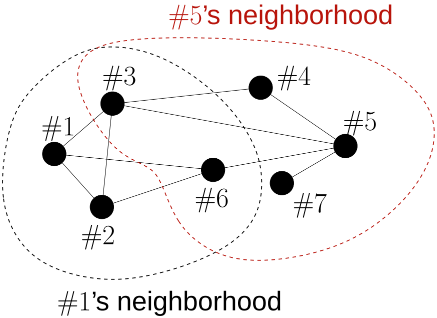
- In a distributed network, the nodes are tasked to collect information and estimate a parameter of interest. The general concept can be summarized as follows:
  - The nodes sense an amount of data from the environment.
  - Computations are performed locally in each node.
  - Each node transmits the locally obtained estimate to a neighborhood of nodes.

Problem Definition

- In a distributed network, the nodes are tasked to collect information and estimate a parameter of interest. The general concept can be summarized as follows:
  - The nodes sense an amount of data from the environment.
  - Computations are performed locally in each node.
  - Each node transmits the locally obtained estimate to a neighborhood of nodes.

The goal is to drive the locally computed estimates to converge to the same value. This is known as consensus.

- The most commonly used topology is the diffusion network:



Problem Formulation

- Let a node set denoted as $\mathcal{N} := \{1, 2, \ldots, N\}$ and each node, k, at time, n, has access to the measurements

$$y_k(n) \in \mathbb{R}, \quad \boldsymbol{x}_{k,n} \in \mathbb{R}^m,$$

Problem Formulation

- Let a node set denoted as $\mathcal{N} := \{1, 2, \ldots, N\}$ and each node, k, at time, n, has access to the measurements

$$y_k(n) \in \mathbb{R}, \quad \boldsymbol{x}_{k,n} \in \mathbb{R}^m,$$

we assume that there exists a linear system, $\boldsymbol{\theta}_*$, such that

$$y_k(n) = \boldsymbol{x}_{k,n}^t \boldsymbol{\theta}_* + v_k(n),$$

where $v_k(n)$ is the noise.

Problem Formulation

- Let a node set denoted as $\mathcal{N} := \{1, 2, \ldots, N\}$ and each node, k, at time, n, has access to the measurements

$$y_k(n) \in \mathbb{R}, \quad \boldsymbol{x}_{k,n} \in \mathbb{R}^m,$$

we assume that there exists a linear system, $\boldsymbol{\theta}_*$, such that

$$y_k(n) = \boldsymbol{x}_{k,n}^t \boldsymbol{\theta}_* + v_k(n),$$

where $v_k(n)$ is the noise.
The task is to estimate the common $\boldsymbol{\theta}_*$.

The Algorithm (node k)

- Combine estimates received from the neighborhood $\mathcal{N}_k$:

$$\phi_k(n) := \sum_{l \in \mathcal{N}_k} c_{k,l}(n+1)\boldsymbol{\theta}_l(n).$$

The Algorithm (node k)

- Combine estimates received from the neighborhood $\mathcal{N}_k$:

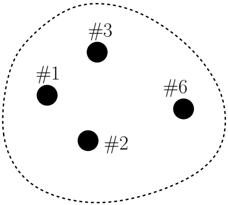
$$\phi_k(n) := \sum_{l \in \mathcal{N}_k} c_{k,l}(n+1)\boldsymbol{\theta}_l(n).$$

- Perform the adaptation step[7]:

$$\boldsymbol{\theta}_k(n+1) := \boldsymbol{\phi}_k(n) + \mu_k(n+1) \left(\sum_{j=n-q+1}^{n} \omega_{k,j} P_{S_{k,j}}(\boldsymbol{\phi}_k(n)) - \boldsymbol{\phi}_k(n) \right).$$

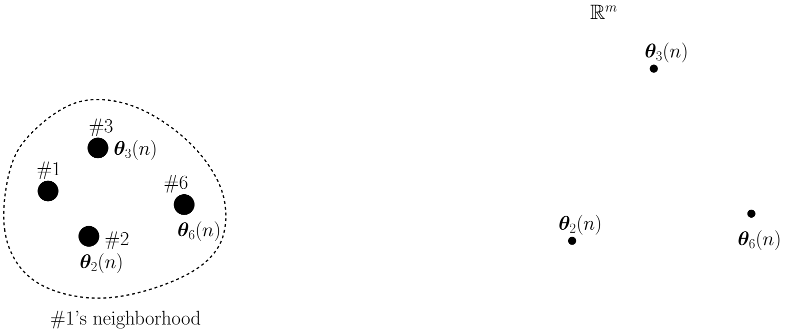
[7] [Chouvardas, Slavakis, Theodoridis, '11].

The Geometry of the Algorithm

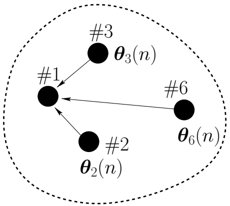


#1's neighborhood

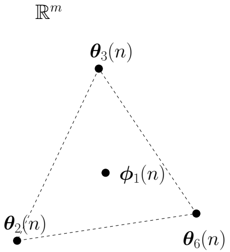
The Geometry of the Algorithm

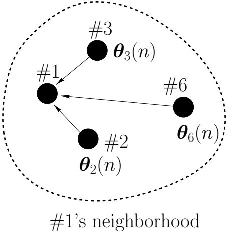


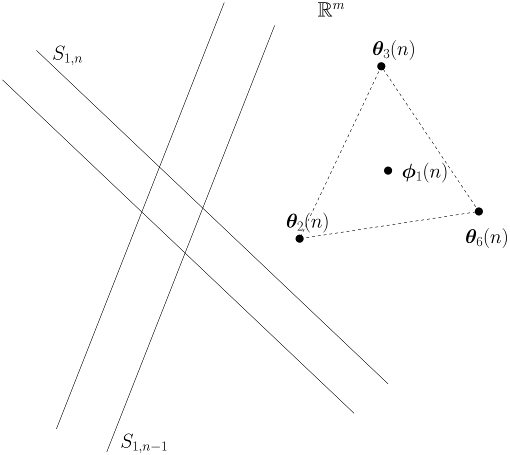
The Geometry of the Algorithm

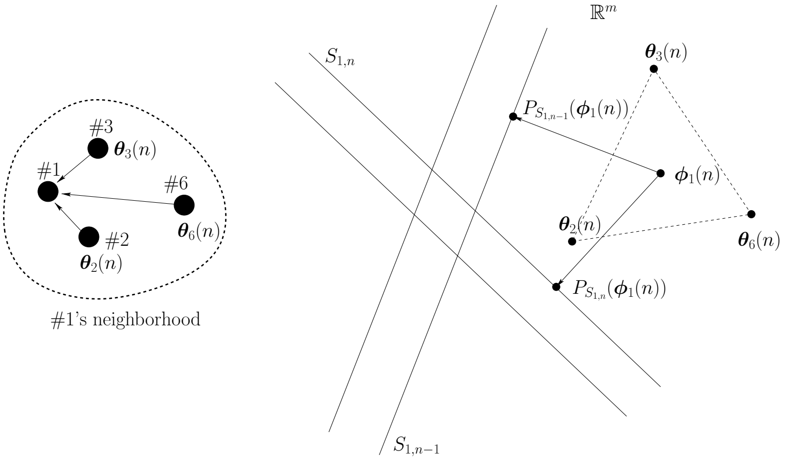


#1's neighborhood

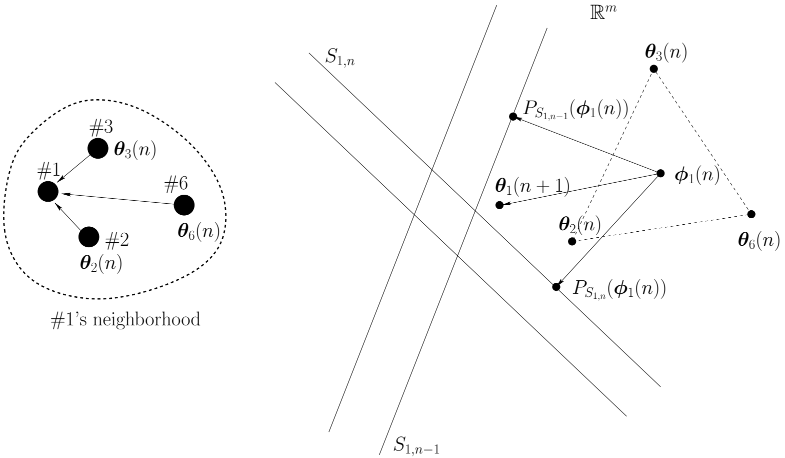



#3
$\boldsymbol{\theta}_3(n)$
#1
#6
#2
$\boldsymbol{\theta}_6(n)$
$\boldsymbol{\theta}_2(n)$
#1's neighborhood


$\mathbb{R}^m$
$S_{1,n}$
$\boldsymbol{\theta}_3(n)$
$\boldsymbol{\phi}_1(n)$
$\boldsymbol{\theta}_2(n)$
$\boldsymbol{\theta}_6(n)$
$S_{1,n-1}$


#3
$\boldsymbol{\theta}_3(n)$
#1
#6
#2
$\boldsymbol{\theta}_6(n)$
$\boldsymbol{\theta}_2(n)$
#1's neighborhood
$\mathbb{R}^m$
$S_{1,n}$
$\boldsymbol{\theta}_3(n)$
$P_{S_{1,n-1}}(\boldsymbol{\phi}_1(n))$
$\boldsymbol{\phi}_1(n)$
$\boldsymbol{\theta}_2(n)$
$\boldsymbol{\theta}_6(n)$
$P_{S_{1,n}}(\boldsymbol{\phi}_1(n))$
$S_{1,n-1}$

The Geometry of the Algorithm



Part B

Outline of Part B

- Incorporate a-priori information into our algorithmic framework.

Outline of Part B

- Incorporate a-priori information into our algorithmic framework.
- An operator theoretic approach will be followed.

Outline of Part B

- Incorporate a-priori information into our algorithmic framework.
- An operator theoretic approach will be followed.
- Such an approach will be illustrated through two paradigms:

Outline of Part B

- Incorporate a-priori information into our algorithmic framework.
- An operator theoretic approach will be followed.
- Such an approach will be illustrated through two paradigms:
  - Beamforming task.

Outline of Part B

- Incorporate a-priori information into our algorithmic framework.
- An operator theoretic approach will be followed.
- Such an approach will be illustrated through two paradigms:
  - Beamforming task.
  - Sparsity-aware learning problem.

Outline of Part B

- Incorporate a-priori information into our algorithmic framework.
- An operator theoretic approach will be followed.
- Such an approach will be illustrated through two paradigms:
  - Beamforming task.
  - Sparsity-aware learning problem.
- Our objective is to show that a large variety of constrained online learning tasks can be unified under a common umbrella; the Adaptive Projected Subgradient Method (APSM).

- A mapping defined in a Hilbert space $\mathcal{H}$:

$$T : \mathcal{H} \to \mathcal{H}.$$

- A mapping defined in a Hilbert space $\mathcal{H}$:

$$T : \mathcal{H} \to \mathcal{H}.$$

- Given a mapping $T : \mathcal{H} \to \mathcal{H}$, its fixed point set is defined as

$$\mathrm{Fix}(T) := \{f \in \mathcal{H} : T(f) = f\},$$

i.e., all those points which are unaffected by T.

- A mapping defined in a Hilbert space $\mathcal{H}$:

$$T : \mathcal{H} \to \mathcal{H}.$$

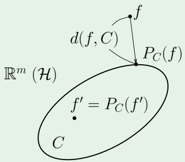
- Given a mapping $T : \mathcal{H} \to \mathcal{H}$, its fixed point set is defined as

$$\mathrm{Fix}(T) := \{f \in \mathcal{H} : T(f) = f\},$$

i.e., all those points which are unaffected by T.

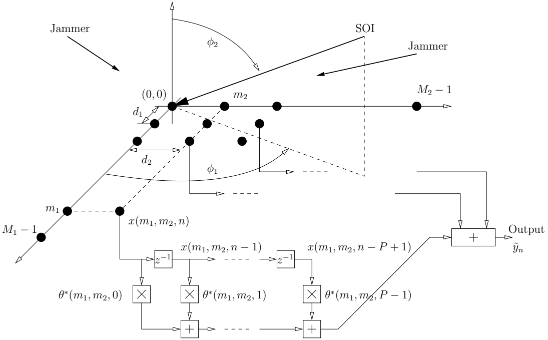
Example

If C is a closed convex set in $\mathcal{H}$, then $\mathrm{Fix}(P_C) = C$.

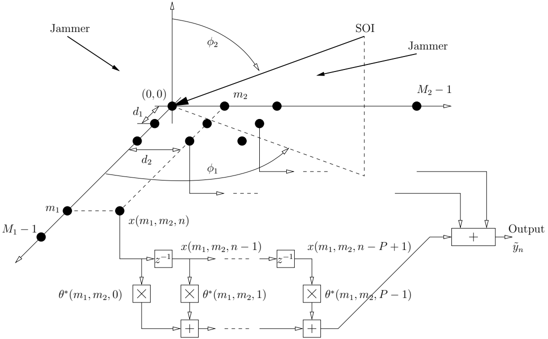


Beamforming

- Antenna arrays are vastly utilized for space-time filtering:



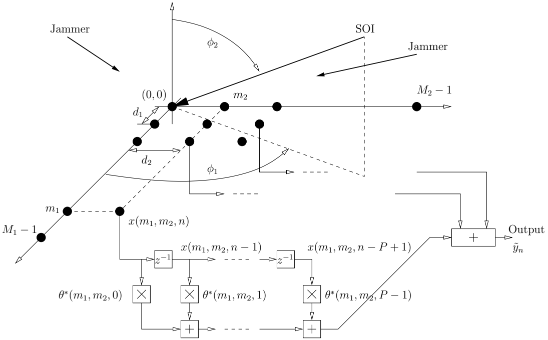
- Antenna arrays are vastly utilized for space-time filtering:



  - The superscript $*$ stands for complex conjugation.

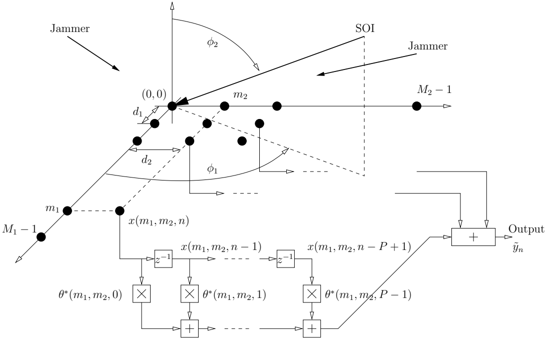
Beamforming

- Antenna arrays are vastly utilized for space-time filtering:



  - The superscript $*$ stands for complex conjugation.
  - SOI: Signal Of Interest.

- Antenna arrays are vastly utilized for space-time filtering:

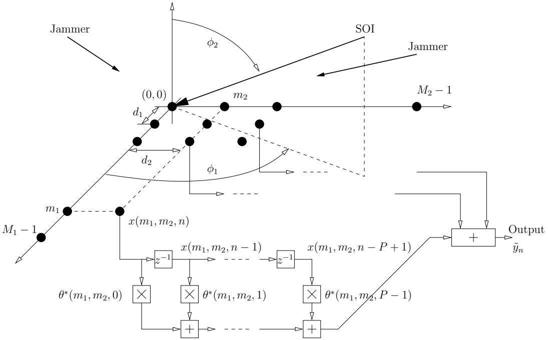


  - The superscript $*$ stands for complex conjugation.
  - SOI: Signal Of Interest.

- After some re-arrangements, the output of the array is given by

$$\tilde{y}_n := \boldsymbol{\theta}^t \boldsymbol{x}_n, \quad n = 0, 1, 2, \ldots.$$

- Antenna arrays are vastly utilized for space-time filtering:



  - The superscript $*$ stands for complex conjugation.
  - SOI: Signal Of Interest.

- After some re-arrangements, the output of the array is given by

$$\tilde{y}_n := \boldsymbol{\theta}^t \boldsymbol{x}_n, \quad n = 0, 1, 2, \ldots .$$

The beamformer is the vector $\boldsymbol{\theta}$.

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

A-priori information

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

A-priori information

- Known locations of the SOIs and/or the jammers.

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

A-priori information

- Known locations of the SOIs and/or the jammers.
- Robustness against erroneous information and array imperfections:

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

A-priori information

- Known locations of the SOIs and/or the jammers.
- Robustness against erroneous information and array imperfections:
  - Knowledge of the approximate location of the SOIs and jammers.

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

A-priori information

- Known locations of the SOIs and/or the jammers.
- Robustness against erroneous information and array imperfections:
  - Knowledge of the approximate location of the SOIs and jammers.
  - Array calibration errors.

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

A-priori information

- Known locations of the SOIs and/or the jammers.
- Robustness against erroneous information and array imperfections:
  - Knowledge of the approximate location of the SOIs and jammers.
  - Array calibration errors.
  - Inoperative array elements.

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

A-priori information

- Known locations of the SOIs and/or the jammers.
- Robustness against erroneous information and array imperfections:
  - Knowledge of the approximate location of the SOIs and jammers.
  - Array calibration errors.
  - Inoperative array elements.
  - Bounds on the weights of the array elements.

The Goal of Beamforming

By utilizing all the available a-priori knowledge, reconstruct the SOIs, while, in the meantime, suppress the jamming signals.

A-priori information

- Known locations of the SOIs and/or the jammers.
- Robustness against erroneous information and array imperfections:
  - Knowledge of the approximate location of the SOIs and jammers.
  - Array calibration errors.
  - Inoperative array elements.
  - Bounds on the weights of the array elements.

Given the previous a-priori info, and the set of data $(y_n, \boldsymbol{x}_n)$, $n = 0, 1, 2, \ldots$, compute $\boldsymbol{\theta}$ such that

$$\boldsymbol{\theta}^t \boldsymbol{x}_n \approx y_n, \quad \forall n.$$

Distortionless and Null Constraints

Definition (Steering vector)

Each transmitting source is associated to a steering vector, $\boldsymbol{s}$, defined as the vector which collects all the signal values in the array if

Distortionless and Null Constraints

Definition (Steering vector)

Each transmitting source is associated to a steering vector, $\boldsymbol{s}$, defined as the vector which collects all the signal values in the array if

- only the source of interest transmits a signal of value 1,

Distortionless and Null Constraints

Definition (Steering vector)

Each transmitting source is associated to a steering vector, $\boldsymbol{s}$, defined as the vector which collects all the signal values in the array if

- only the source of interest transmits a signal of value 1,
- and there is no noise in the system.

Distortionless and Null Constraints

Definition (Steering vector)

Each transmitting source is associated to a steering vector, $\boldsymbol{s}$, defined as the vector which collects all the signal values in the array if

- only the source of interest transmits a signal of value 1,
- and there is no noise in the system.

Remark: The steering vector comprises information like the location of the associated source, and the geometry of the array.

Distortionless and Null Constraints

Definition (Steering vector)

Each transmitting source is associated to a steering vector, $\boldsymbol{s}$, defined as the vector which collects all the signal values in the array if

- only the source of interest transmits a signal of value 1,
- and there is no noise in the system.

Remark: The steering vector comprises information like the location of the associated source, and the geometry of the array.

Distortionless constraint

If $\boldsymbol{s}_{\text{SOI}}$ is the steering vector associated to a SOI, then we would like to have:

$$\boldsymbol{s}_{\text{SOI}}^t \boldsymbol{\theta} = 1.$$

Distortionless and Null Constraints

Definition (Steering vector)

Each transmitting source is associated to a steering vector, $\boldsymbol{s}$, defined as the vector which collects all the signal values in the array if

- only the source of interest transmits a signal of value 1,
- and there is no noise in the system.

Remark: The steering vector comprises information like the location of the associated source, and the geometry of the array.

Distortionless constraint

If $\boldsymbol{s}_{\text{SOI}}$ is the steering vector associated to a SOI, then we would like to have:

$$\boldsymbol{s}_{\text{SOI}}^t \boldsymbol{\theta} = 1.$$

Nulls

If $\boldsymbol{s}_{\text{jam}}$ is the steering vector associated to a jammer, then we would like to have:

$$\boldsymbol{s}_{\text{jam}}^t \boldsymbol{\theta} = 0.$$

Affinely Constrained Beamforming

A large variety of a-priori knowledge in beamforming problems can be cast by means of affine constraints; given a matrix C and a vector $\boldsymbol{g}$:

$$\boldsymbol{C}^t \boldsymbol{\theta} = \boldsymbol{g}.$$

Affinely Constrained Beamforming

A large variety of a-priori knowledge in beamforming problems can be cast by means of affine constraints; given a matrix $\boldsymbol{C}$ and a vector $\boldsymbol{g}$:

$$\boldsymbol{C}^t \boldsymbol{\theta} = \boldsymbol{g}.$$

Example

Let $\boldsymbol{C} := [\boldsymbol{s}_{\text{SOI}}, \boldsymbol{s}_{\text{jam}}]$, and $\boldsymbol{g} := [1, 0]^t$.

Affinely Constrained Beamforming

A large variety of a-priori knowledge in beamforming problems can be cast by means of affine constraints; given a matrix C and a vector $\boldsymbol{g}$:

$$\boldsymbol{C}^t \boldsymbol{\theta} = \boldsymbol{g}.$$

Example

Let $\boldsymbol{C} := [\boldsymbol{s}_{\text{SOI}}, \boldsymbol{s}_{\text{jam}}]$, and $\boldsymbol{g} := [1, 0]^t$.

Define the following affine set $V := \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \left\|\boldsymbol{C}^t\boldsymbol{\theta} - \boldsymbol{g}\right\|$,

Affinely Constrained Beamforming

A large variety of a-priori knowledge in beamforming problems can be cast by means of affine constraints; given a matrix $\boldsymbol{C}$ and a vector $\boldsymbol{g}$:

$$\boldsymbol{C}^t\boldsymbol{\theta} = \boldsymbol{g}.$$

Example

Let $\boldsymbol{C} := [\boldsymbol{s}_{\text{SOI}}, \boldsymbol{s}_{\text{jam}}]$, and $\boldsymbol{g} := [1, 0]^t$.

Define the following affine set $V := \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \left\|\boldsymbol{C}^t\boldsymbol{\theta} - \boldsymbol{g}\right\|$, which contains, in general, an infinite number of points, and covers also the case of inconsistent a-priori constraints, i.e., the case:

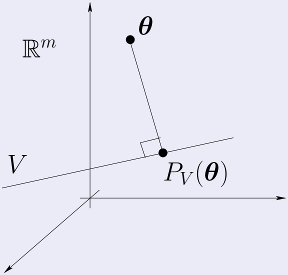
$$\forall\boldsymbol{\theta}, \quad \boldsymbol{C}^t\boldsymbol{\theta} \neq \boldsymbol{g}.$$

Projection onto the affine set V

Given $V := \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^m} \left\|C^t\boldsymbol{\theta} - \boldsymbol{g}\right\|$, the metric projection mapping onto V is given by

$$P_V(\boldsymbol{\theta}) = \boldsymbol{\theta} - C^{t\dagger}\left(C^t\boldsymbol{\theta} - \boldsymbol{g}\right), \quad \forall\boldsymbol{\theta} \in \mathbb{R}^m,$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse of a matrix.



Affinely Constrained Algorithm

- At time n, given the training data $(y_n, \boldsymbol{x}_n)$, define the hyperslab:

$$S_n[\epsilon] := \left\{ \boldsymbol{\theta} \in \mathbb{R}^m : \left| \boldsymbol{x}_n^t \boldsymbol{\theta} - y_n \right| \le \epsilon \right\}.$$

Affinely Constrained Algorithm

- At time n, given the training data $(y_n, \boldsymbol{x}_n)$, define the hyperslab:

$$S_n[\epsilon] := \left\{\boldsymbol{\theta} \in \mathbb{R}^m : \left|\boldsymbol{x}_n^t \boldsymbol{\theta} - y_n\right| \leq \epsilon\right\}.$$

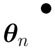
- For any initial point $\boldsymbol{\theta}_0$, and $\forall n$,

$$\boldsymbol{\theta}_{n+1} := P_V\left(\boldsymbol{\theta}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} P_{S_i[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\right)\right),$$

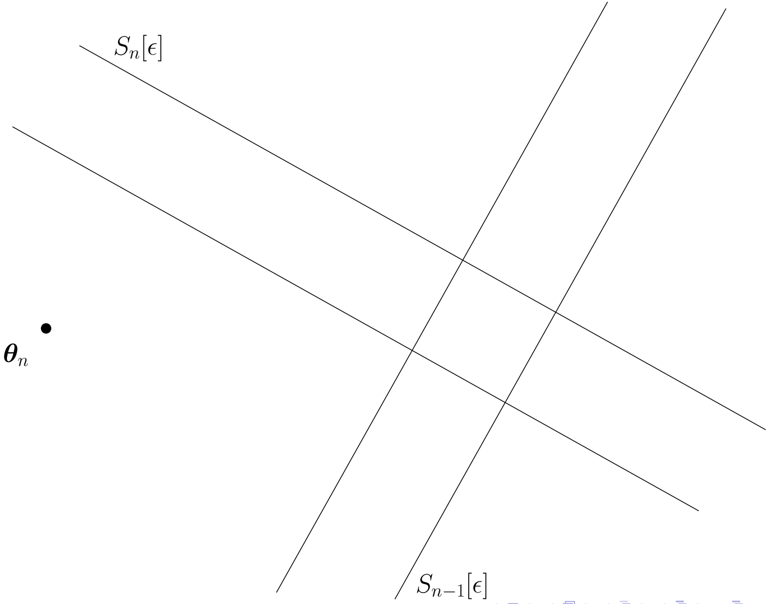
$$\mu_n \in (0, 2\mathcal{M}_n),$$

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \|P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\|^2}{\|\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

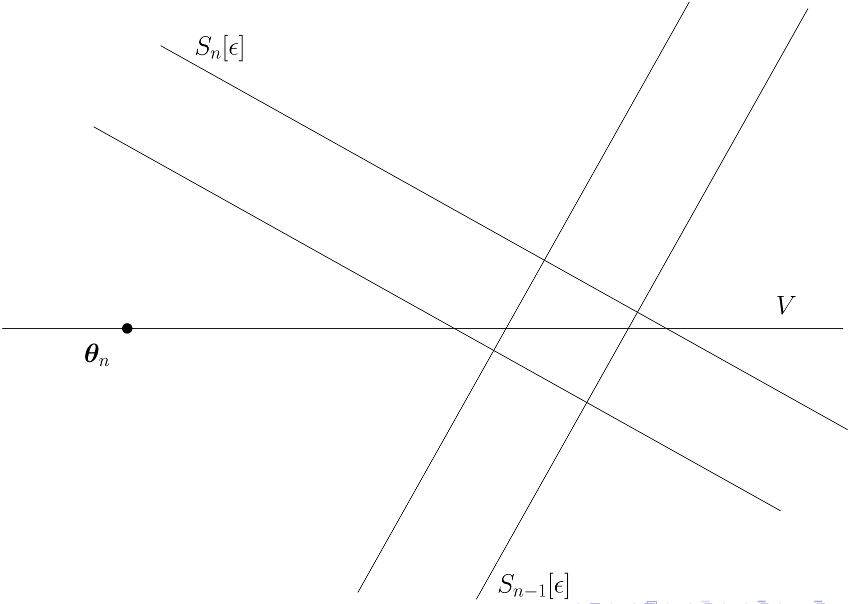
Geometry of the Algorithm



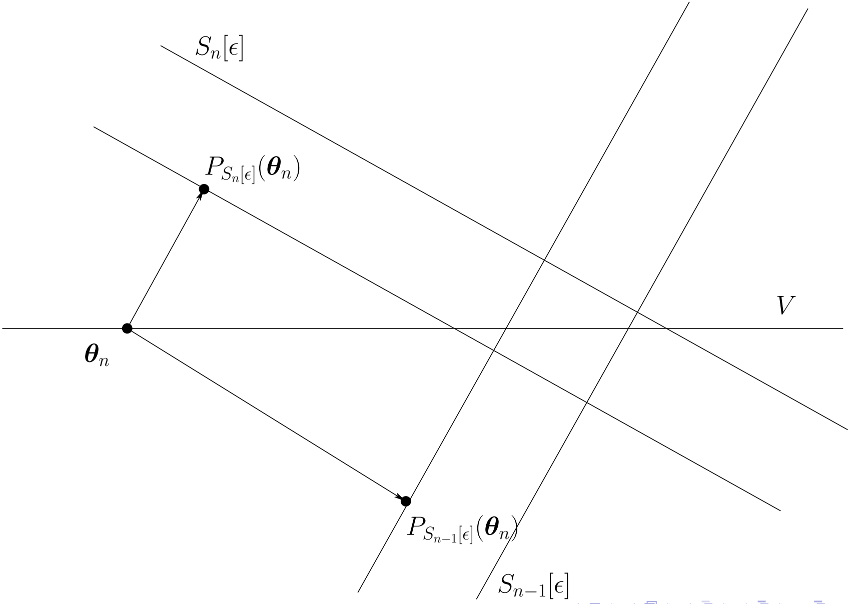
Geometry of the Algorithm



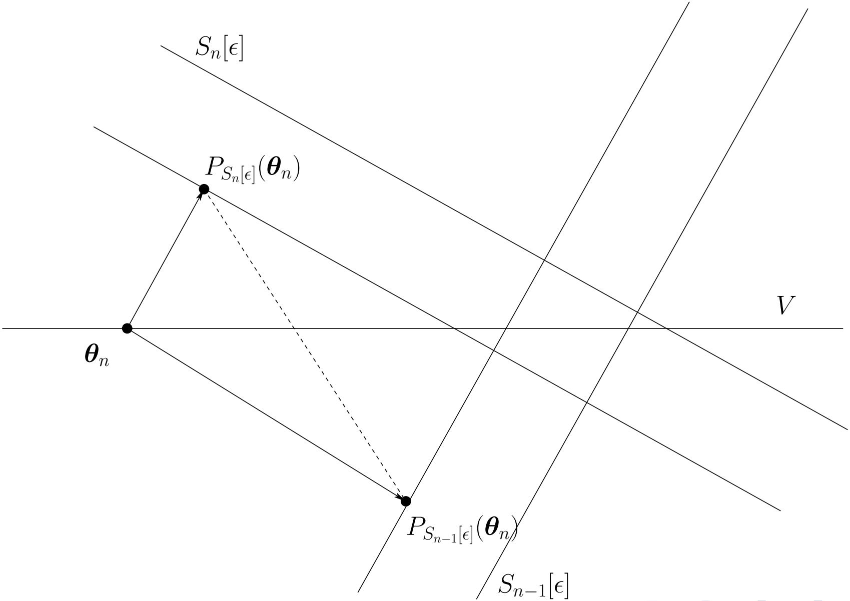
Geometry of the Algorithm



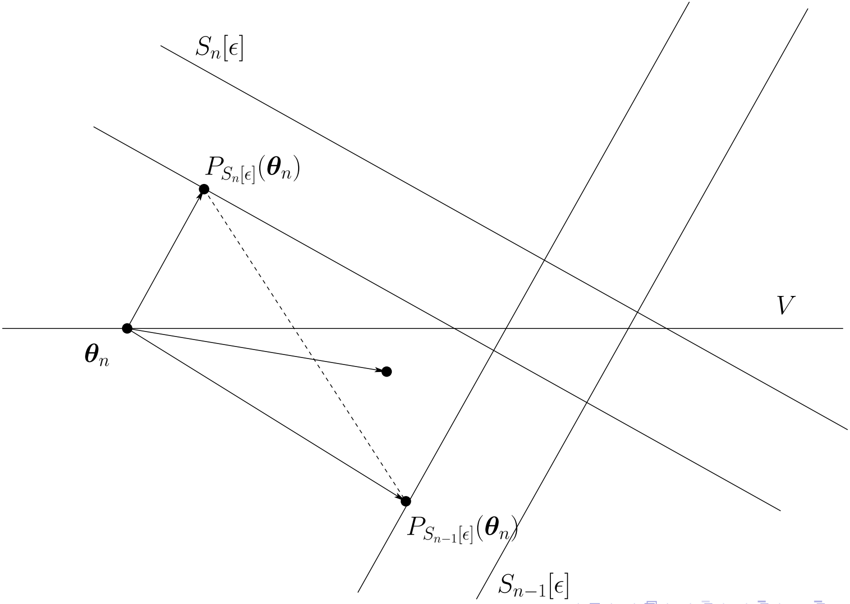
Geometry of the Algorithm

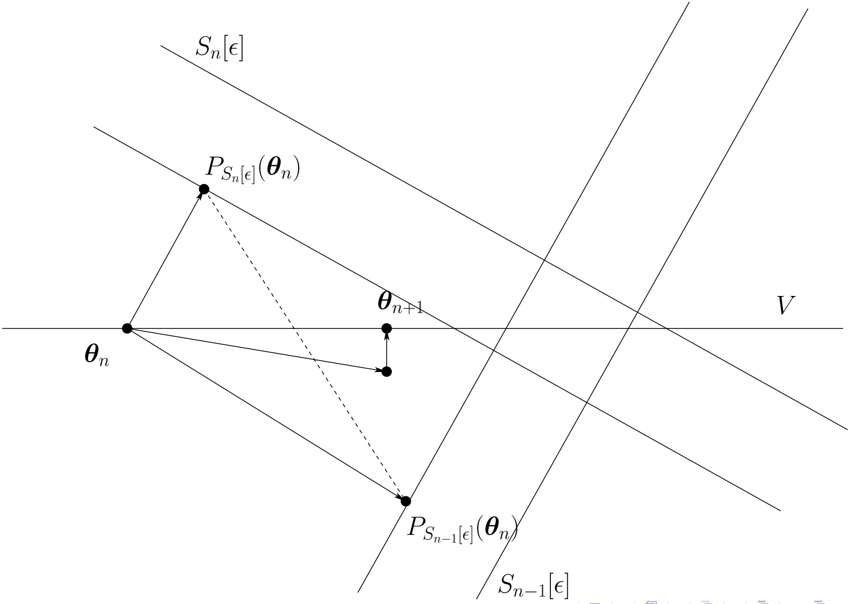


Geometry of the Algorithm



Geometry of the Algorithm




$S_n[\epsilon]$
$P_{S_n[\epsilon]}(\boldsymbol{\theta}_n)$
$\boldsymbol{\theta}_{n+1}$
V
$\boldsymbol{\theta}_n$
$P_{S_{n-1}[\epsilon]}(\boldsymbol{\theta}_n)$
$S_{n-1}[\epsilon]$

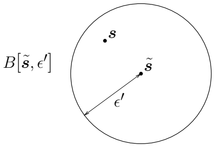
- Robustness is a key design issue in beamforming.

- Robustness is a key design issue in beamforming.
- There are cases, for example, where the location of the SOI is known approximately.

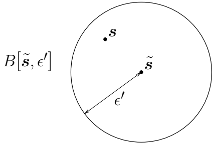
- Robustness is a key design issue in beamforming.
- There are cases, for example, where the location of the SOI is known approximately.
- A mathematical formulation for such a scenario is as follows;

- Robustness is a key design issue in beamforming.
- There are cases, for example, where the location of the SOI is known approximately.
- A mathematical formulation for such a scenario is as follows;
  - given the approximate steering vector $\tilde{s}$,

- Robustness is a key design issue in beamforming.
- There are cases, for example, where the location of the SOI is known approximately.
- A mathematical formulation for such a scenario is as follows;
  - given the approximate steering vector $\tilde{\boldsymbol{s}}$,
  - and a ball of uncertainty $B[\tilde{\boldsymbol{s}}, \epsilon']$, of radius ϵ' around $\tilde{\boldsymbol{s}}$:



- Robustness is a key design issue in beamforming.
- There are cases, for example, where the location of the SOI is known approximately.
- A mathematical formulation for such a scenario is as follows;
  - given the approximate steering vector $\tilde{\boldsymbol{s}}$,
  - and a ball of uncertainty $B[\tilde{\boldsymbol{s}}, \epsilon']$, of radius ϵ' around $\tilde{\boldsymbol{s}}$:



  - calculate those $\boldsymbol{\theta}$ such that, for some user-defined $\epsilon'' \geq 0$,

$$\boldsymbol{\theta}^t \boldsymbol{s} \in [1 - \epsilon'', 1 + \epsilon''], \quad \forall \boldsymbol{s} \in B[\tilde{\boldsymbol{s}}, \epsilon'].$$

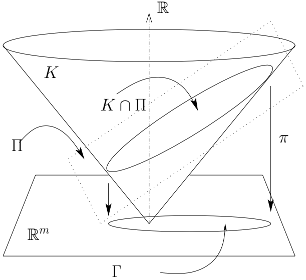
The Icecream Cone

- The previous task breaks down to a number of more fundamental problems of the following type; find a vector that belongs to

$$\Gamma := \{\boldsymbol{\theta} \in \mathbb{R}^m : \boldsymbol{\theta}^t \boldsymbol{s} \geq \gamma, \forall \boldsymbol{s} \in B[\tilde{\boldsymbol{s}}, \epsilon']\} = \left\{ \begin{array}{c} \text{all vectors that satisfy an} \\ \text{infinite number of inequalities} \end{array} \right\}.$$

- If $\Gamma \neq \emptyset$, then the previous problem is equivalent to[8]

finding a point in $K \cap \Pi$,
K: an icecream cone,
Π: a hyperplane.



[8] [Slavakis, Yamada' 07], [Slavakis, Theodoridis, Yamada '09].

The Complete Picture

Given $(\boldsymbol{x}_n, y_n)$, find a $\boldsymbol{\theta} \in \mathbb{R}^m$ such that

$$\left|\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n\right| \leq \epsilon,$$

Given $(\boldsymbol{x}_n, y_n)$, find a $\boldsymbol{\theta} \in \mathbb{R}^m$ such that

$$\left|\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n\right| \leq \epsilon,$$
$$\boldsymbol{\theta}^t \boldsymbol{s} \geq \gamma,\ \forall \boldsymbol{s} \in B[\tilde{\boldsymbol{s}}, \epsilon'], \qquad \text{(Robustness)}.$$

Algorithm for Robust Regression

Assume weights $\omega_j^{(n)} \geq 0$ such that $\sum_{j=n-q+1}^{n} \omega_j^{(n)} = 1$. For any $\boldsymbol{\theta}_0 \in \mathbb{R}^m$,

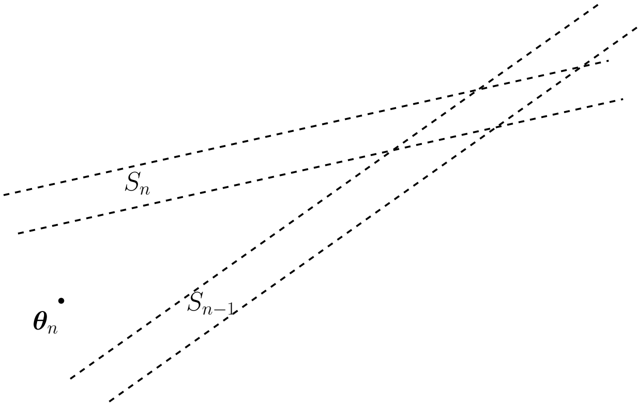
$$\boldsymbol{\theta}_{n+1} := P_{\Pi} P_K \left(\boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n \right) \right), \quad \forall n \geq 0,$$

where the extrapolation coefficient $\mu_n \in (0, 2\mathcal{M}_n)$ with

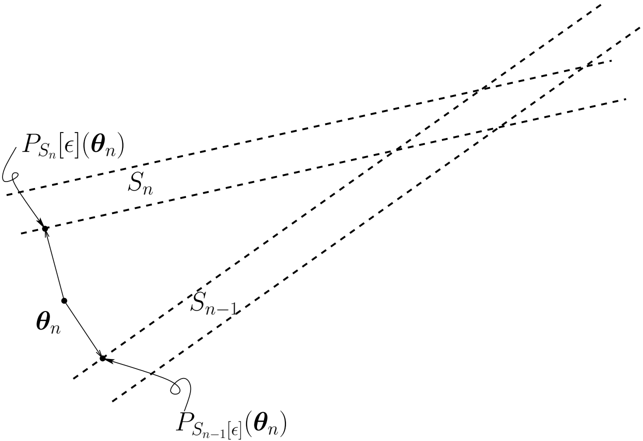
$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \|P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\|^2}{\|\sum_{j=n-q+1} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\|^2}, & \text{if } \sum_{j=n-q+1} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

Geometry of the Algorithm

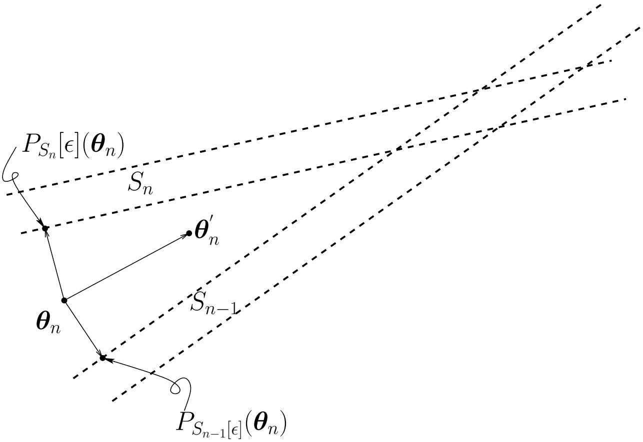
Geometry of the Algorithm



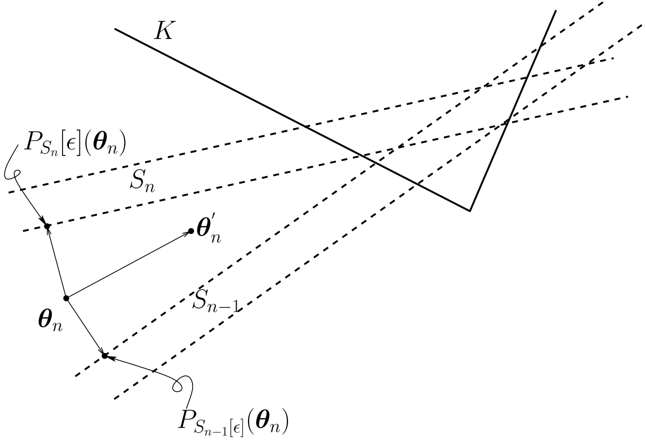
Geometry of the Algorithm



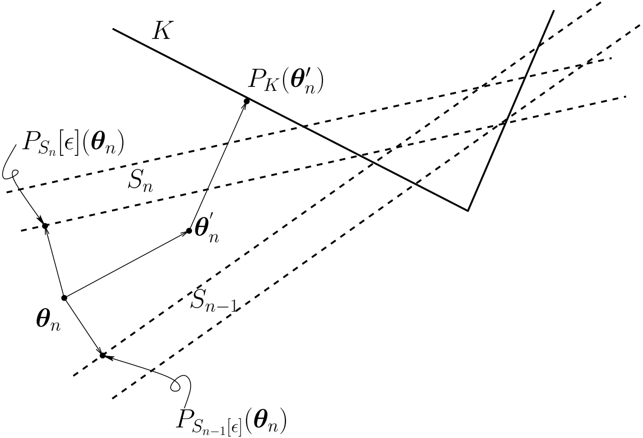
Geometry of the Algorithm



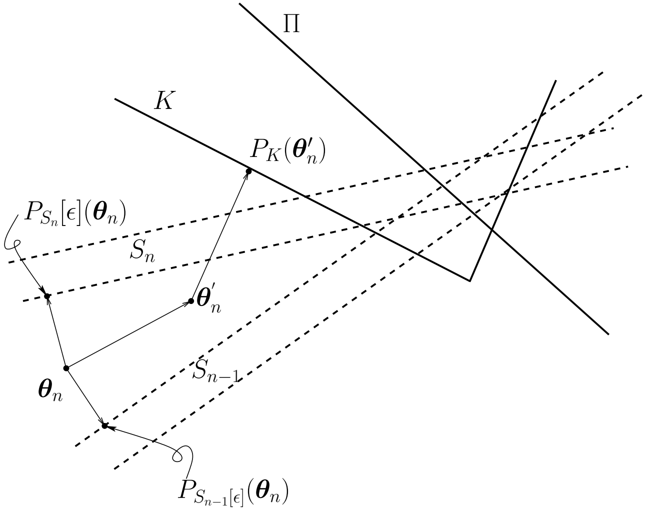
Geometry of the Algorithm

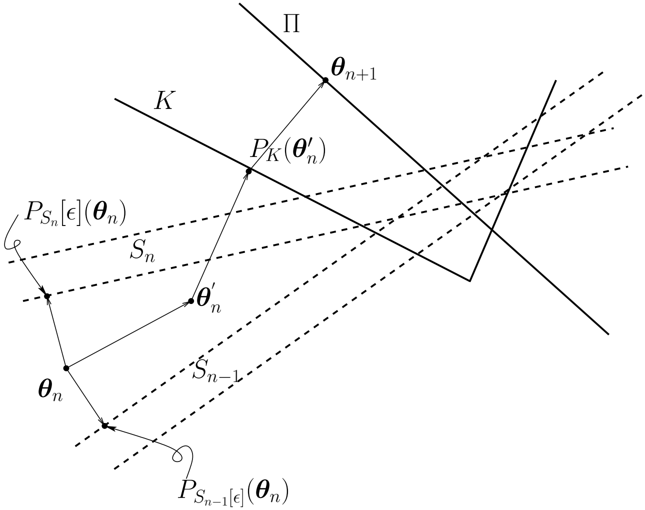


Geometry of the Algorithm



Geometry of the Algorithm




Π
$\boldsymbol{\theta}_{n+1}$
K
$P_K(\boldsymbol{\theta}_n')$
$P_{S_n}[\epsilon](\boldsymbol{\theta}_n)$
S_n
$\boldsymbol{\theta}_n'$
S_{n-1}
$\boldsymbol{\theta}_n$
$P_{S_{n-1}[\epsilon]}(\boldsymbol{\theta}_n)$

Handling A-Priori Information

How did we handle a-priori information?

Handling A-Priori Information

How did we handle a-priori information?

- Each piece of a-priori info was represented by a closed convex set, e.g., K, Π.

Handling A-Priori Information

How did we handle a-priori information?

- Each piece of a-priori info was represented by a closed convex set, e.g., K, Π.
- In order to be in agreement with all of the pieces of the a-priori info, we looked for a point into the intersection of the closed convex sets, e.g., $K \cap \Pi$.

Handling A-Priori Information

How did we handle a-priori information?

- Each piece of a-priori info was represented by a closed convex set, e.g., K, Π.
- In order to be in agreement with all of the pieces of the a-priori info, we looked for a point into the intersection of the closed convex sets, e.g., $K \cap \Pi$.
- In algorithmic terms, we utilized the composition mapping $P_\Pi P_K : \mathbb{R}^m \to \mathbb{R}^m$.

Handling A-Priori Information

How did we handle a-priori information?

- Each piece of a-priori info was represented by a closed convex set, e.g., K, Π.
- In order to be in agreement with all of the pieces of the a-priori info, we looked for a point into the intersection of the closed convex sets, e.g., $K \cap \Pi$.
- In algorithmic terms, we utilized the composition mapping $P_\Pi P_K : \mathbb{R}^m \to \mathbb{R}^m$.

This strategy reminds us of POCS:

POCS

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^p C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. Then,

$$\forall \boldsymbol{\theta} \in \mathbb{R}^m, \quad \left(P_{C_p} \cdots P_{C_1}\right)^n (\boldsymbol{\theta}) \xrightarrow[n\to\infty]{w} {}^{\exists}\boldsymbol{\theta}_* \in \bigcap_{i=1}^p C_i.$$

Handling A-Priori Information

How did we handle a-priori information?

- Each piece of a-priori info was represented by a closed convex set, e.g., K, Π.
- In order to be in agreement with all of the pieces of the a-priori info, we looked for a point into the intersection of the closed convex sets, e.g., $K \cap \Pi$.
- In algorithmic terms, we utilized the composition mapping $P_\Pi P_K : \mathbb{R}^m \to \mathbb{R}^m$.

This strategy reminds us of POCS:

POCS

Given a finite number of closed convex sets $C_1, \ldots, C_p$, with $\bigcap_{i=1}^p C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_p}$. Then,

$$\forall \boldsymbol{\theta} \in \mathbb{R}^m, \quad \left(P_{C_p} \cdots P_{C_1}\right)^n (\boldsymbol{\theta}) \xrightarrow[n\to\infty]{w} {}^{\exists}\boldsymbol{\theta}_* \in \bigcap_{i=1}^{p} C_i.$$

Key assumption

The a-priori info is consistent, i.e., $\bigcap_{i=1}^p C_i \neq \emptyset$.

Inconsistent A-Priori Information

- Is the case of inconsistent a-priori info possible in practice?

Inconsistent A-Priori Information

- Is the case of inconsistent a-priori info possible in practice?
- Yes, in highly constrained learning tasks; the more constraints we add to the problem, the smaller the intersection of the associated convex sets becomes.

Inconsistent A-Priori Information

- Is the case of inconsistent a-priori info possible in practice?
- Yes, in highly constrained learning tasks; the more constraints we add to the problem, the smaller the intersection of the associated convex sets becomes.

Example

A beamforming problem where there is erroneous info on SOI and jammer steering vectors, array calibration errors, info on inoperative array elements, and stringent bounds on the weights of the array.

Inconsistent A-Priori Information

- Is the case of inconsistent a-priori info possible in practice?
- Yes, in highly constrained learning tasks; the more constraints we add to the problem, the smaller the intersection of the associated convex sets becomes.

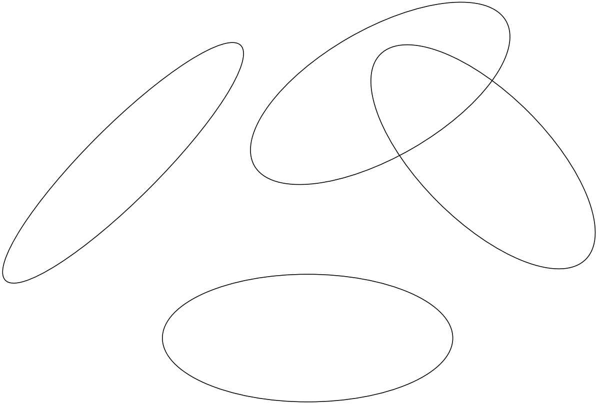
Example

A beamforming problem where there is erroneous info on SOI and jammer steering vectors, array calibration errors, info on inoperative array elements, and stringent bounds on the weights of the array.

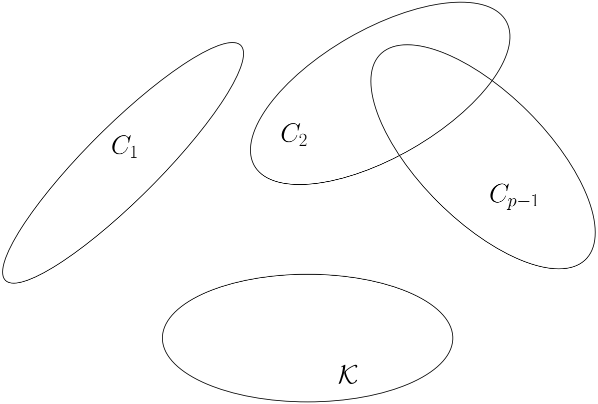
How do we deal with the case of inconsistent a-priori info, i.e.,

$$\bigcap_{i=1}^{p} C_i = \emptyset?$$

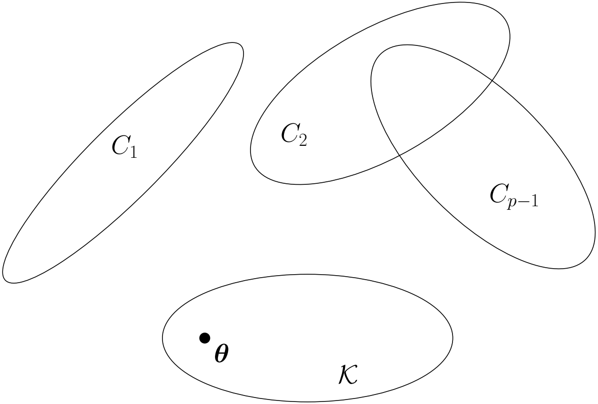
An Intuitive Suggestion



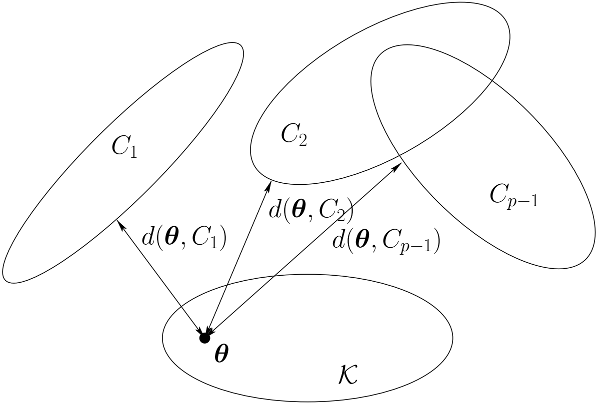
An Intuitive Suggestion



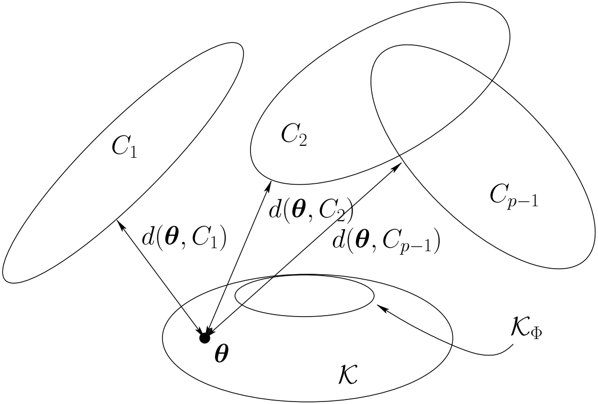
An Intuitive Suggestion

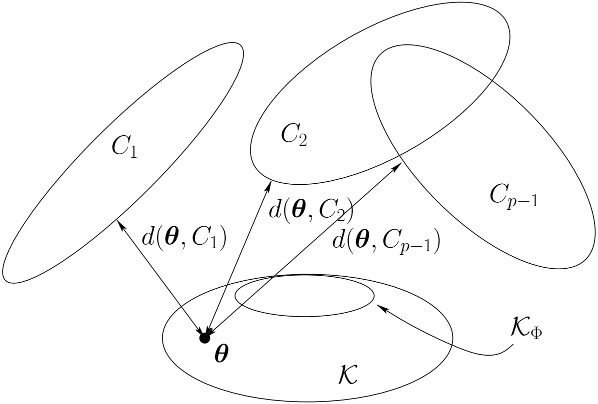


An Intuitive Suggestion



An Intuitive Suggestion





Definition ($\mathcal{K}_\Phi$)

All those points of $\mathcal{K}$ which minimize a function Φ of the distances $\{d(\cdot, C_i)\}_{i=1}^{p-1}$.

A Way to Handle Inconsistent A-Priori Information

Given a number of a-priori constraints, represented as p closed convex sets,

A Way to Handle Inconsistent A-Priori Information

Given a number of a-priori constraints, represented as p closed convex sets,

- Identify one of them as the absolute constraint $\mathcal{K}$, and rename the other ones as $C_1, C_2, \ldots, C_{p-1}$.

A Way to Handle Inconsistent A-Priori Information

Given a number of a-priori constraints, represented as p closed convex sets,

- Identify one of them as the absolute constraint $\mathcal{K}$, and rename the other ones as $C_1, C_2, \ldots, C_{p-1}$.
- Assign to each C_i a convex weight β_i, i.e., $\beta_i \in (0, 1]$ and $\sum_{i=1}^{p-1} \beta_i = 1$.

A Way to Handle Inconsistent A-Priori Information

Given a number of a-priori constraints, represented as p closed convex sets,

- Identify one of them as the absolute constraint $\mathcal{K}$, and rename the other ones as $C_1, C_2, \ldots, C_{p-1}$.
- Assign to each C_i a convex weight β_i, i.e., $\beta_i \in (0, 1]$ and $\sum_{i=1}^{p-1} \beta_i = 1$.
- Define the function:

$$\Phi(\boldsymbol{\theta}) := \frac{1}{2} \sum_{i=1}^{p-1} \beta_i d^2(\boldsymbol{\theta}, C_i), \quad \forall \boldsymbol{\theta} \in \mathcal{K}.$$

Our objective is to look for the minimizers $\mathcal{K}_\Phi$ of this function.

A Way to Handle Inconsistent A-Priori Information

Given a number of a-priori constraints, represented as p closed convex sets,

- Identify one of them as the absolute constraint $\mathcal{K}$, and rename the other ones as $C_1, C_2, \ldots, C_{p-1}$.
- Assign to each C_i a convex weight β_i, i.e., $\beta_i \in (0,1]$ and $\sum_{i=1}^{p-1} \beta_i = 1$.
- Define the function:

$$\Phi(\boldsymbol{\theta}) := \frac{1}{2} \sum_{i=1}^{p-1} \beta_i d^2(\boldsymbol{\theta}, C_i), \quad \forall \boldsymbol{\theta} \in \mathcal{K}.$$

Our objective is to look for the minimizers $\mathcal{K}_\Phi$ of this function.

Notice that $\Phi' = I - \sum_{i=1}^{p-1} \beta_i P_{C_i}$.

A Way to Handle Inconsistent A-Priori Information

Given a number of a-priori constraints, represented as p closed convex sets,

- Identify one of them as the absolute constraint $\mathcal{K}$, and rename the other ones as $C_1, C_2, \ldots, C_{p-1}$.
- Assign to each C_i a convex weight β_i, i.e., $\beta_i \in (0, 1]$ and $\sum_{i=1}^{p-1} \beta_i = 1$.
- Define the function:

$$\Phi(\boldsymbol{\theta}) := \frac{1}{2} \sum_{i=1}^{p-1} \beta_i d^2(\boldsymbol{\theta}, C_i), \quad \forall \boldsymbol{\theta} \in \mathcal{K}.$$

Our objective is to look for the minimizers $\mathcal{K}_\Phi$ of this function.

Notice that $\Phi' = I - \sum_{i=1}^{p-1} \beta_i P_{C_i}$.

- Define the mapping $T : \mathbb{R}^m \to \mathbb{R}^m$ as

$$T := P_{\mathcal{K}} \left(I - \lambda \left(I - \sum_{i=1}^{p-1} \beta_i P_{C_i} \right) \right), \quad \lambda \in (0, 2).$$

A Way to Handle Inconsistent A-Priori Information

Given a number of a-priori constraints, represented as p closed convex sets,

- Identify one of them as the absolute constraint $\mathcal{K}$, and rename the other ones as $C_1, C_2, \ldots, C_{p-1}$.
- Assign to each C_i a convex weight β_i, i.e., $\beta_i \in (0, 1]$ and $\sum_{i=1}^{p-1} \beta_i = 1$.
- Define the function:

$$\Phi(\boldsymbol{\theta}) := \frac{1}{2} \sum_{i=1}^{p-1} \beta_i d^2(\boldsymbol{\theta}, C_i), \quad \forall \boldsymbol{\theta} \in \mathcal{K}.$$

  Our objective is to look for the minimizers $\mathcal{K}_\Phi$ of this function.

Notice that $\Phi' = I - \sum_{i=1}^{p-1} \beta_i P_{C_i}$.

- Define the mapping $T : \mathbb{R}^m \to \mathbb{R}^m$ as

$$T := P_{\mathcal{K}} \left(I - \lambda \left(I - \sum_{i=1}^{p-1} \beta_i P_{C_i} \right) \right), \quad \lambda \in (0, 2).$$

- Then, $\mathrm{Fix}(T) = \mathcal{K}_\Phi$.

The Algorithm

For any $\boldsymbol{\theta}_0 \in \mathbb{R}^m$,

$$\boldsymbol{\theta}_{n+1} := T\left(\boldsymbol{\theta}_n + \mu_n\left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\right)\right), \quad \forall n \geq 0,$$

where the extrapolation coefficient $\mu_n \in (0, 2\mathcal{M}_n)$ with

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \|P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\|^2}{\|\sum_{j=n-q+1} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\|^2}, & \text{if } \sum_{j=n-q+1} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) \neq \boldsymbol{\theta}_n, \\ 1, & \text{otherwise.} \end{cases}$$

Problem definition

Problem definition

- In a number of applications, many of the parameters to be estimated are a-priori known to be zero. That is, the parameter vector, $\boldsymbol{\theta}$, is sparse.

$$\boldsymbol{\theta}^t = [*, *, 0, 0, 0, *, 0, \ldots].$$

Problem definition

- In a number of applications, many of the parameters to be estimated are a-priori known to be zero. That is, the parameter vector, $\boldsymbol{\theta}$, is sparse.

$$\boldsymbol{\theta}^t = [*, *, 0, 0, 0, *, 0, \ldots].$$

  If the locations of the zeros were known, the problem would be trivial.

  However, the locations of the zeros are not known a-priori. This makes the task challenging.

Problem definition

- In a number of applications, many of the parameters to be estimated are a-priori known to be zero. That is, the parameter vector, $\boldsymbol{\theta}$, is sparse.

$$\boldsymbol{\theta}^t = [*, *, 0, 0, 0, *, 0, \ldots].$$

  If the locations of the zeros were known, the problem would be trivial.

  However, the locations of the zeros are not known a-priori. This makes the task challenging.

- Typical applications include echo cancellation in Internet telephony, MIMO channel estimation, Compressed Sensing (CS), etc.

Problem definition

- In a number of applications, many of the parameters to be estimated are a-priori known to be zero. That is, the parameter vector, $\boldsymbol{\theta}$, is sparse.

$$\boldsymbol{\theta}^t = [*, *, 0, 0, 0, *, 0, \ldots].$$

If the locations of the zeros were known, the problem would be trivial.

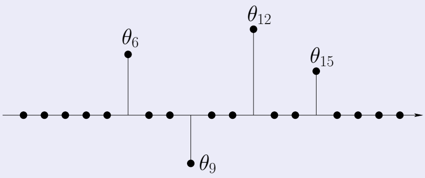
However, the locations of the zeros are not known a-priori. This makes the task challenging.

- Typical applications include echo cancellation in Internet telephony, MIMO channel estimation, Compressed Sensing (CS), etc.
- Sparsity promotion is achieved via ℓ_1-norm regularization of a loss function:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \sum_{n=0}^{N} \mathcal{L}\left(y_n, \boldsymbol{x}_n^t \boldsymbol{\theta}\right) + \lambda \left\|\boldsymbol{\theta}\right\|_1, \quad \lambda > 0.$$

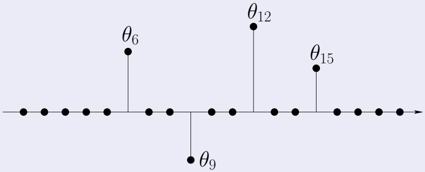
The ℓ_0 norm

$$\|\boldsymbol{\theta}\|_0 := \text{card}\{i : \theta_i \neq 0\}.$$



The ℓ_0 norm

$$\|\boldsymbol{\theta}\|_0 := \operatorname{card}\{i : \theta_i \neq 0\}.$$



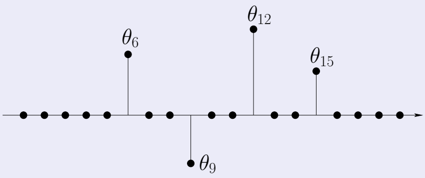
- Consider the linear model:

$$y_n := \boldsymbol{x}_n^t \boldsymbol{\theta} + v_n, \quad \forall n,$$

where $(v_n)_{n \geq 0}$ denotes the noise process.

The ℓ_0 norm

$$\|\boldsymbol{\theta}\|_0 := \text{card}\{i : \theta_i \neq 0\}.$$



- Consider the linear model:

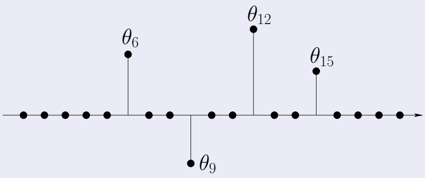
$$y_n := \boldsymbol{x}_n^t \boldsymbol{\theta} + v_n, \quad \forall n,$$

where $(v_n)_{n \geq 0}$ denotes the noise process.

- Define $\boldsymbol{X}_N := [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, $\boldsymbol{y}_N := [y_0, y_1, \ldots, y_N]^t$, and $\epsilon \geq 0$.

The ℓ_0 norm

$$\|\boldsymbol{\theta}\|_0 := \text{card}\big\{i : \theta_i \neq 0\big\}.$$



- Consider the linear model:

$$y_n := \boldsymbol{x}_n^t \boldsymbol{\theta} + v_n, \quad \forall n,$$

  where $(v_n)_{n \geq 0}$ denotes the noise process.
- Define $\boldsymbol{X}_N := [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, $\boldsymbol{y}_N := [y_0, y_1, \ldots, y_N]^t$, and $\epsilon \geq 0$.
- A typical Compressed Sensing task is formulated as follows:

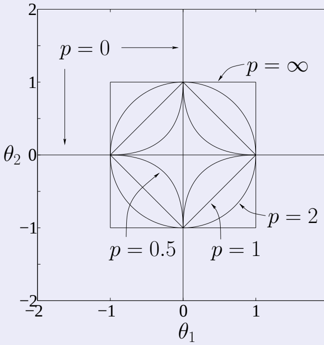
$$\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \|\boldsymbol{\theta}\|_0$$

$$\text{s.t.} \quad \left\|\boldsymbol{X}_N^t \boldsymbol{\theta} - \boldsymbol{y}_N\right\| \leq \epsilon.$$

Alternatives to the ℓ_0 Norm

The ℓ_p norm $(0 < p \leq 1)$

$$\|\boldsymbol{\theta}\|_p := \left(\sum_{i=1}^{m} |\theta_i|^p \right)^{\frac{1}{p}}.$$



The ℓ_1-ball case

The ℓ_1-ball case

- Given $(\boldsymbol{x}_n, y_n)$, $n = 0, 1, 2, \ldots$, find $\boldsymbol{\theta}$ such that

$$\left|\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n\right| \leq \epsilon, \quad n = 0, 1, 2, \ldots$$
$$\boldsymbol{\theta} \in B_{\ell_1}[\delta] := \left\{\boldsymbol{\theta}' \in \mathbb{R}^m : \left\|\boldsymbol{\theta}'\right\|_1 \leq \delta\right\}.$$

Algorithm for Sparsity-Aware Learning

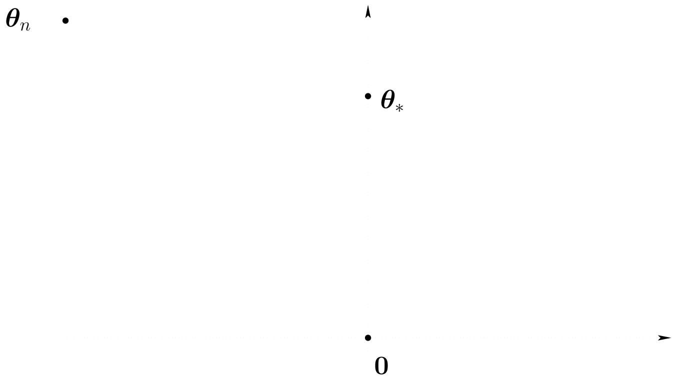
The ℓ_1-ball case

- Given $(\boldsymbol{x}_n, y_n)$, $n = 0, 1, 2, \ldots$, find $\boldsymbol{\theta}$ such that

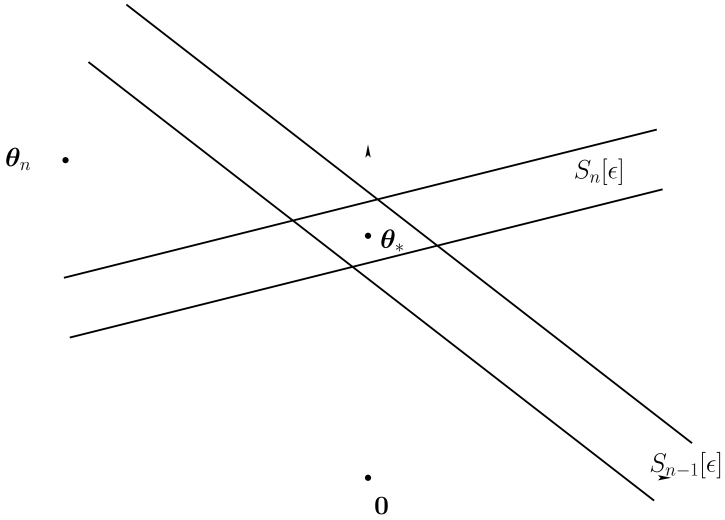
$$\left|\boldsymbol{\theta}^t \boldsymbol{x}_n - y_n\right| \leq \epsilon, \quad n = 0, 1, 2, \ldots$$
$$\boldsymbol{\theta} \in B_{\ell_1}[\delta] := \left\{\boldsymbol{\theta}' \in \mathbb{R}^m : \left\|\boldsymbol{\theta}'\right\|_1 \leq \delta\right\}.$$

- The recursion:

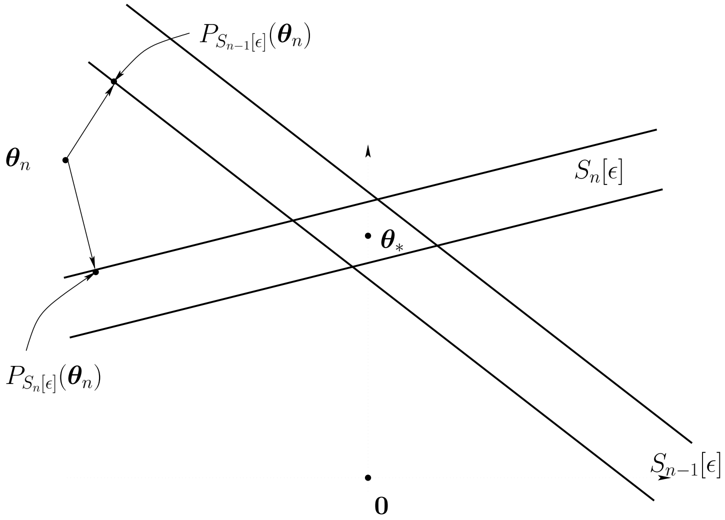
$$\boldsymbol{\theta}_{n+1} := P_{B_{\ell_1}[\delta]}\left(\boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\right)\right).$$


$\boldsymbol{\theta}_n$
$\boldsymbol{\theta}_*$
$\mathbf{0}$

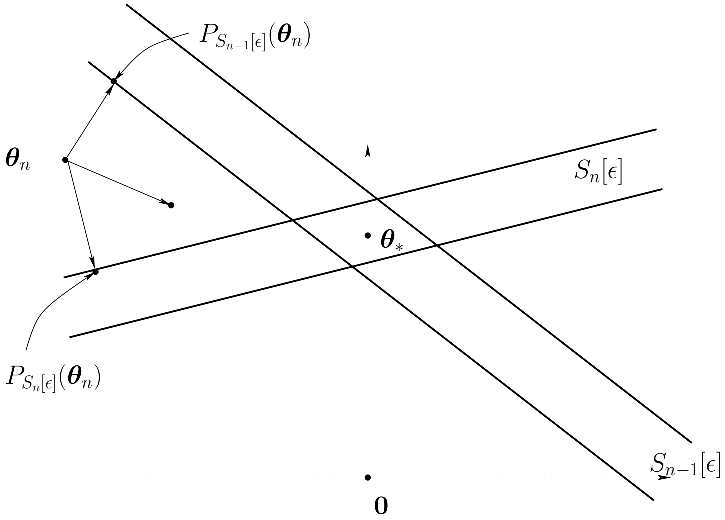
Geometric Illustration of the Algorithm



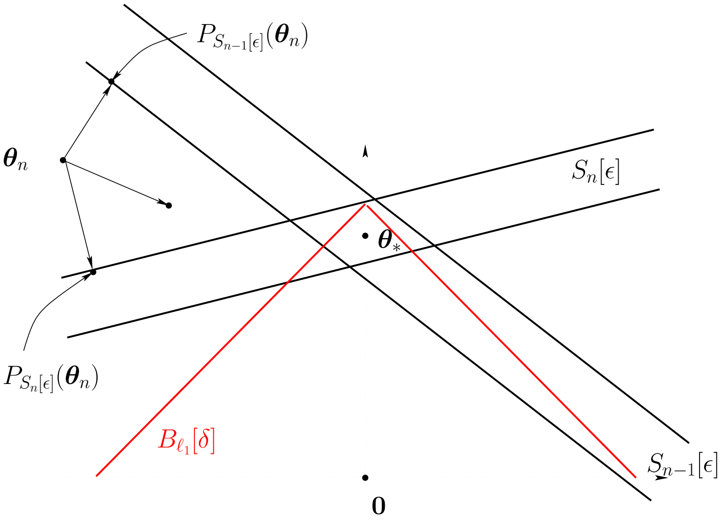
Geometric Illustration of the Algorithm



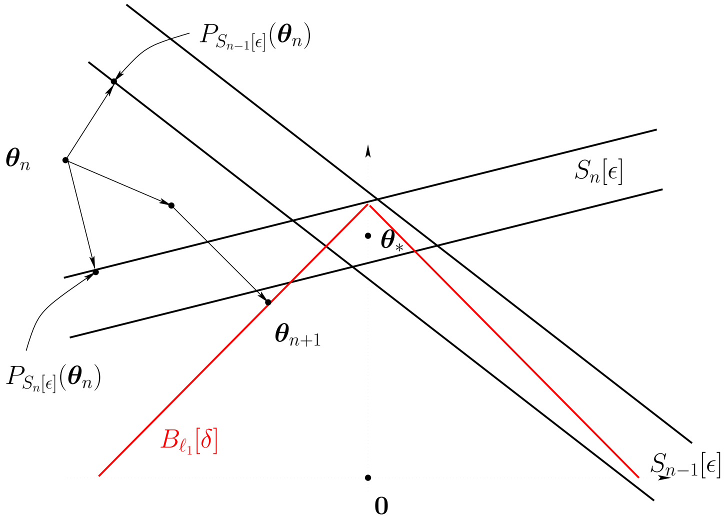
Geometric Illustration of the Algorithm



Geometric Illustration of the Algorithm



Geometric Illustration of the Algorithm



Remark: The convergence can be significantly speeded up, if in place of the ℓ_1-ball a weighted ℓ_1-ball is used to constrain the solutions.

Remark: The convergence can be significantly speeded up, if in place of the ℓ_1-ball a weighted ℓ_1-ball is used to constrain the solutions.

- Definition:

$$\|\boldsymbol{\theta}\|_{1,w} := \sum_{i=1}^{m} w_i |\theta_i|,$$

$$B_{\ell_1}[\boldsymbol{w}_n, \delta] := \left\{\boldsymbol{\theta} \in \mathbb{R}^m : \|\boldsymbol{\theta}\|_{1,w} \leq \delta\right\}.$$

Remark: The convergence can be significantly speeded up, if in place of the ℓ_1-ball a weighted ℓ_1-ball is used to constrain the solutions.

- Definition:

$$\|\boldsymbol{\theta}\|_{1,w} := \sum_{i=1}^{m} w_i |\theta_i|,$$

$$B_{\ell_1}[\boldsymbol{w}_n, \delta] := \left\{\boldsymbol{\theta} \in \mathbb{R}^m : \|\boldsymbol{\theta}\|_{1,w} \leq \delta\right\}.$$

- Time-adaptive weighted norm:

$$w_{n,i} := \frac{1}{|\theta_{n,i}| + \epsilon'_n}.$$

Remark: The convergence can be significantly speeded up, if in place of the ℓ_1-ball a weighted ℓ_1-ball is used to constrain the solutions.

- Definition:

$$\|\boldsymbol{\theta}\|_{1,w} := \sum_{i=1}^{m} w_i |\theta_i|,$$

$$B_{\ell_1}[\boldsymbol{w}_n, \delta] := \left\{\boldsymbol{\theta} \in \mathbb{R}^m : \|\boldsymbol{\theta}\|_{1,w} \leq \delta\right\}.$$

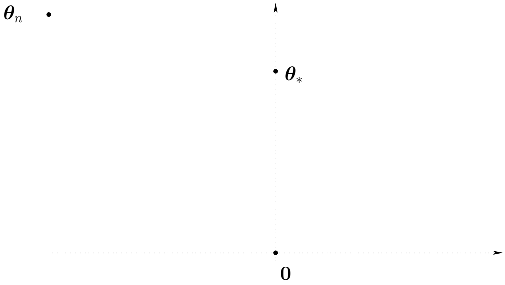
- Time-adaptive weighted norm:

$$w_{n,i} := \frac{1}{|\theta_{n,i}| + \epsilon'_n}.$$

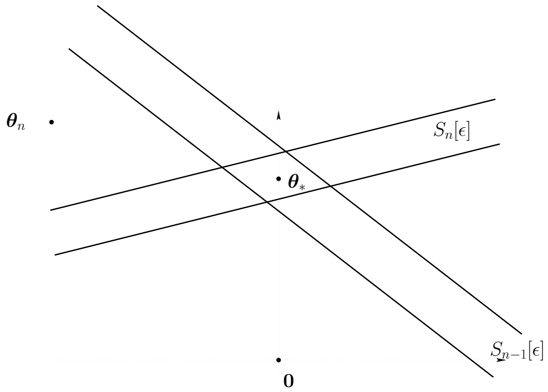
- The recursion[9]:

$$\boldsymbol{\theta}_{n+1} := P_{B_{\ell_1}[\boldsymbol{w}_n,\delta]} \left(\boldsymbol{\theta}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{\theta}_n) - \boldsymbol{\theta}_n\right)\right).$$

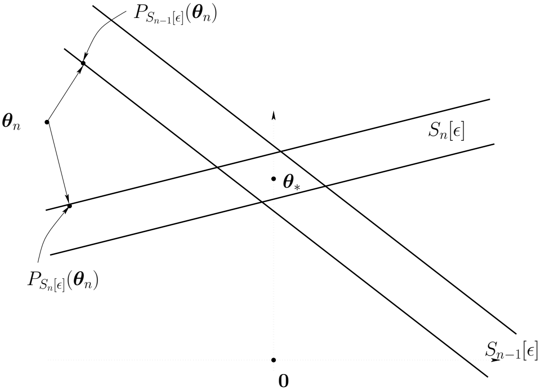
[9] [Kopsinis, Slavakis, Theodoridis, '11].


θ_n
θ_*
0

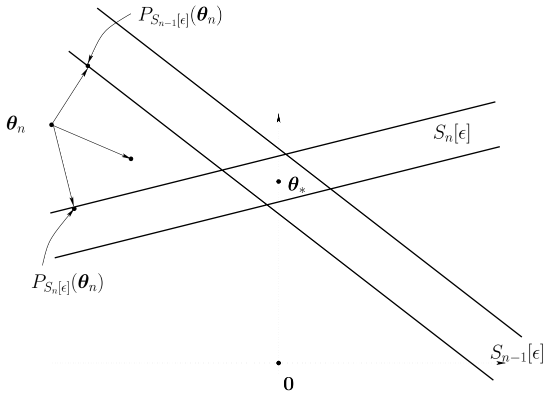
Geometric Illustration of the Algorithm



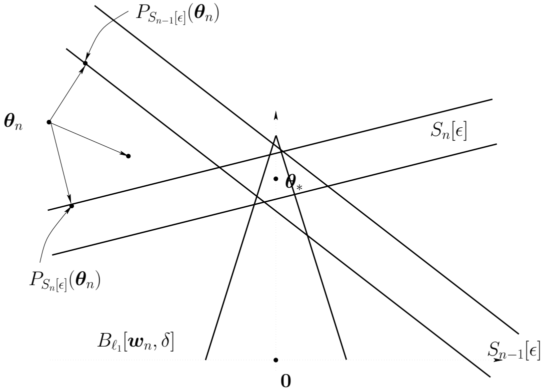
Geometric Illustration of the Algorithm



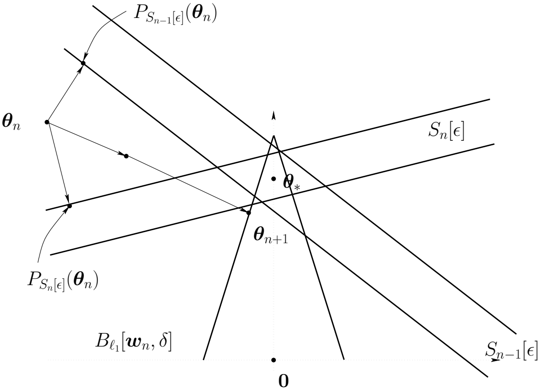
Geometric Illustration of the Algorithm



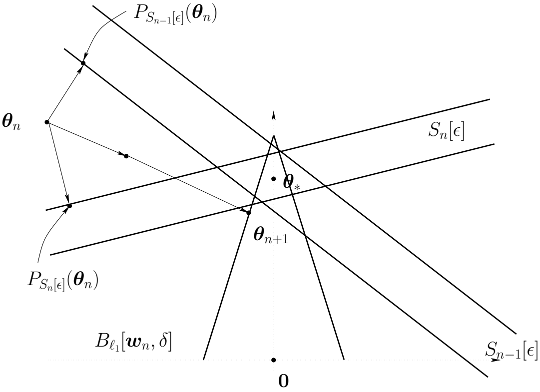
Geometric Illustration of the Algorithm



Geometric Illustration of the Algorithm



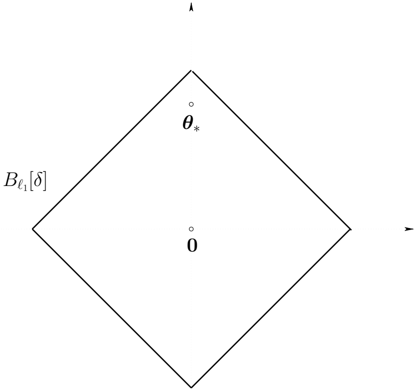
Geometric Illustration of the Algorithm



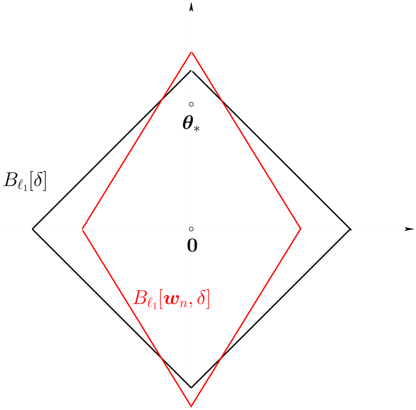
Projecting onto $B_{\ell_1}[\boldsymbol{w}_n, \delta]$ is equivalent to a specific soft thresholding operation.

Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.

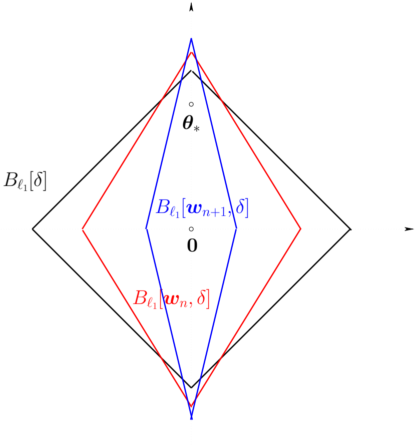
Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.



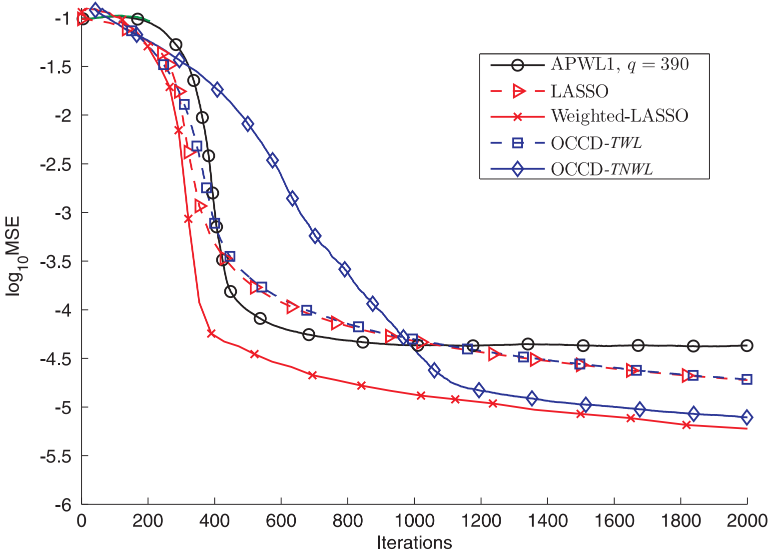
Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.



Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.

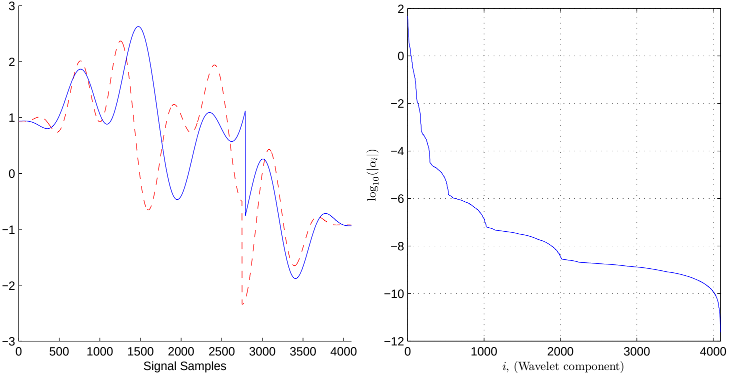


Time Invariant Signal



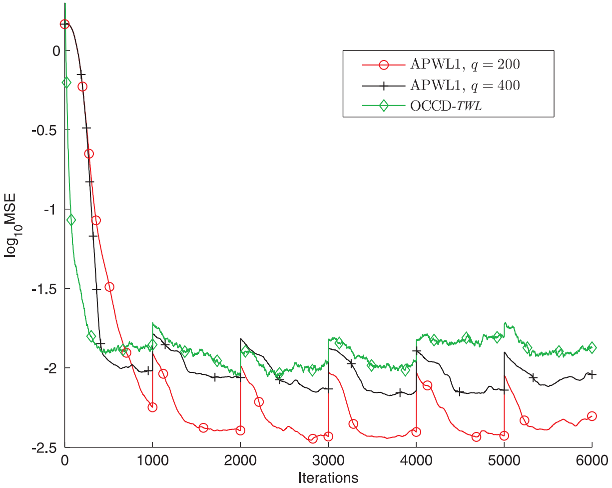
$m := 1024$, $\|\boldsymbol{\theta}_*\|_0 := 100$ wavelet coefficients. The radius of the ℓ_1-ball is set to $\delta := 101$.

Time Varying Signal



$m := 4096$. The radius of the ℓ_1-ball is set to $\delta := 40$.
The sum of two chirp signals.

Time Varying Signal



Movies of the OCCD, and the APWL1sub.

Generalized Thresholding

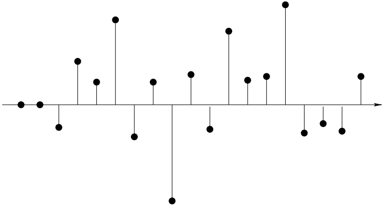
What we have seen so far corresponds to soft thresholding operations.

Generalized Thresholding

What we have seen so far corresponds to soft thresholding operations.

Hard thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.

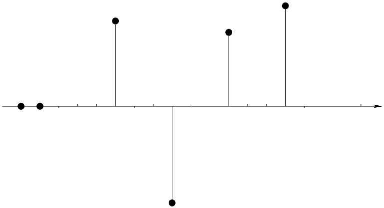


Generalized Thresholding

What we have seen so far corresponds to soft thresholding operations.

Hard thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Keep those as they are, while nullify the rest of them.

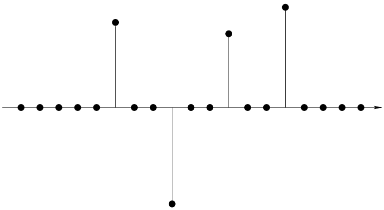


Generalized Thresholding

What we have seen so far corresponds to soft thresholding operations.

Hard thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Keep those as they are, while nullify the rest of them.

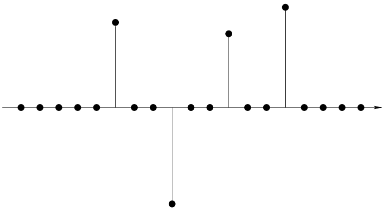


Generalized Thresholding

What we have seen so far corresponds to soft thresholding operations.

Hard thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Keep those as they are, while nullify the rest of them.



Generalized thresholding

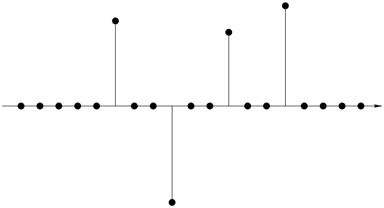
- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.

Generalized Thresholding

What we have seen so far corresponds to soft thresholding operations.

Hard thresholding

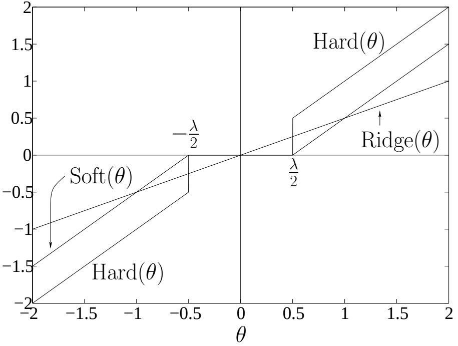
- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Keep those as they are, while nullify the rest of them.



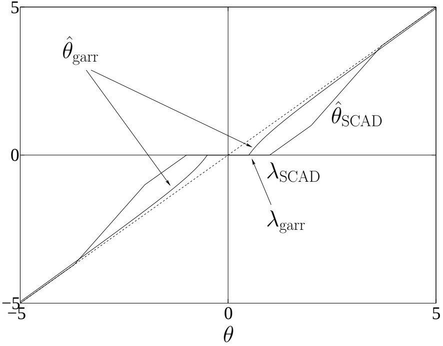
Generalized thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Shrink, under some rule, the rest of the components.

Examples of Generalized Thresholding Mappings



(a) Hard, soft thresholding, and the ridge regression estimate.



(b) The SCAD and garrote thresholding.

Penalized Least-Squares Thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.

Mathematical Formulation of Thresholding

Penalized Least-Squares Thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Let θ_i be one of the rest of the components.

Mathematical Formulation of Thresholding

Penalized Least-Squares Thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Let θ_i be one of the rest of the components.
- In order to shrink θ_i, solve the optimization task:

$$\min_{\hat{\theta}_i \in \mathbb{R}} \frac{1}{2}\left(\hat{\theta}_i - \theta_i\right)^2 + \lambda p\left(\left|\hat{\theta}_i\right|\right), \quad \lambda > 0,$$

where $p(\cdot)$ stands for a user-defined penalty function, which might be non-convex.

Mathematical Formulation of Thresholding

Penalized Least-Squares Thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Let θ_i be one of the rest of the components.
- In order to shrink θ_i, solve the optimization task:

$$\min_{\hat{\theta}_i \in \mathbb{R}} \frac{1}{2}\left(\hat{\theta}_i - \theta_i\right)^2 + \lambda p\left(\left|\hat{\theta}_i\right|\right), \quad \lambda > 0,$$

  where $p(\cdot)$ stands for a user-defined penalty function, which might be non-convex.
- Under some mild conditions, the previous optimization task possesses a unique solution $\hat{\theta}_{i*}$.

Mathematical Formulation of Thresholding

Penalized Least-Squares Thresholding

- Identify the K largest, in magnitude, components of a vector $\boldsymbol{\theta}$.
- Let θ_i be one of the rest of the components.
- In order to shrink θ_i, solve the optimization task:

$$\min_{\hat{\theta}_i \in \mathbb{R}} \frac{1}{2}\left(\hat{\theta}_i - \theta_i\right)^2 + \lambda p\left(\left|\hat{\theta}_i\right|\right), \quad \lambda > 0,$$

  where $p(\cdot)$ stands for a user-defined penalty function, which might be non-convex.

- Under some mild conditions, the previous optimization task possesses a unique solution $\hat{\theta}_{i*}$.

Definition (Generalized Thresholding Mapping)

The Generalized Thresholding mapping is defined as follows:

$$T_{\mathrm{GT}} : \theta_i \mapsto \hat{\theta}_{i*}.$$

Fixed Point Set of T_{GT}

- Given K, define the set of all tuples of length K:

$$\mathscr{T} := \{(i_1, i_2, \ldots, i_K) : 1 \leq i_1 < i_2 < \ldots < i_K \leq m\}.$$

Fixed Point Set of T_{GT}

- Given K, define the set of all tuples of length K:

$$\mathscr{T} := \{(i_1, i_2, \ldots, i_K) : 1 \leq i_1 < i_2 < \ldots < i_K \leq m\}.$$

- Given a tuple $J \in \mathscr{T}$, define the subspace:

$$M_J := \left\{\boldsymbol{\theta} \in \mathbb{R}^m : \theta_i = 0, \forall i \notin J\right\}.$$

Fixed Point Set of T_{GT}

- Given K, define the set of all tuples of length K:

$$\mathscr{T} := \{(i_1, i_2, \ldots, i_K) : 1 \leq i_1 < i_2 < \ldots < i_K \leq m\}.$$

- Given a tuple $J \in \mathscr{T}$, define the subspace:

$$M_J := \left\{\boldsymbol{\theta} \in \mathbb{R}^m : \theta_i = 0, \forall i \notin J\right\}.$$

- Then, the fixed point set of T_{GT} is a union of subspaces:

$$\mathrm{Fix}\big(T_{\mathrm{GT}}\big) = \bigcup_{J \in \mathscr{T}} M_J, \quad \text{(non-convex set)}.$$

Fixed Point Set of T_{GT}

- Given K, define the set of all tuples of length K:

$$\mathscr{T} := \{(i_1, i_2, \ldots, i_K) : 1 \leq i_1 < i_2 < \ldots < i_K \leq m\}.$$

- Given a tuple $J \in \mathscr{T}$, define the subspace:

$$M_J := \left\{\boldsymbol{\theta} \in \mathbb{R}^m : \theta_i = 0, \forall i \notin J\right\}.$$

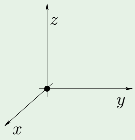
- Then, the fixed point set of T_{GT} is a union of subspaces:

$$\mathrm{Fix}(T_{\mathsf{GT}}) = \bigcup_{J \in \mathscr{T}} M_J, \quad \text{(non-convex set)}.$$

Example

For the 3-dimensional case $\mathbb{R}^3$, and if $K := 2$,

$$\mathrm{Fix}(T_{\mathsf{GT}}) = \begin{array}{c} xy\text{-plane} \cup yz\text{-plane} \\ \cup\, xz\text{-plane}. \end{array}$$



Definition (Nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$ is called nonexpansive if

$$\|T(f_1) - T(f_2)\| \leq \|f_1 - f_2\|, \qquad \forall f_1, f_2 \in \mathcal{H}.$$

The fixed point set of a nonexpansive mapping is closed and convex.

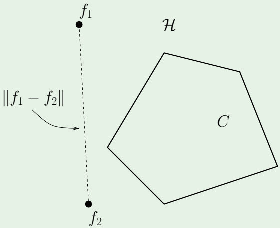
Definition (Nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$ is called nonexpansive if

$$\|T(f_1) - T(f_2)\| \leq \|f_1 - f_2\|, \qquad \forall f_1, f_2 \in \mathcal{H}.$$

The fixed point set of a nonexpansive mapping is closed and convex.

Example (Projection Mapping)



First Steps Towards a Unifying Framework

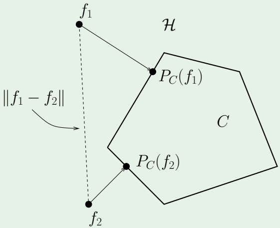
Definition (Nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$ is called nonexpansive if

$$\|T(f_1) - T(f_2)\| \leq \|f_1 - f_2\|, \qquad \forall f_1, f_2 \in \mathcal{H}.$$

The fixed point set of a nonexpansive mapping is closed and convex.

Example (Projection Mapping)



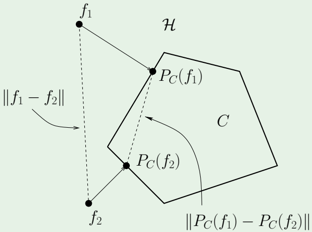
Definition (Nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$ is called nonexpansive if

$$\|T(f_1) - T(f_2)\| \leq \|f_1 - f_2\|, \qquad \forall f_1, f_2 \in \mathcal{H}.$$

The fixed point set of a nonexpansive mapping is closed and convex.

Example (Projection Mapping)



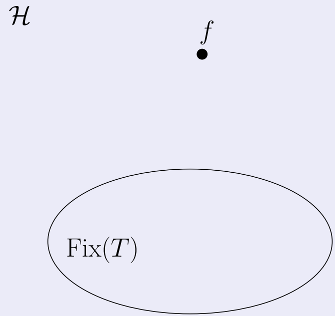
$$\mathrm{Fix}(P_C) = C.$$

Definition (Quasi-nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$, with $\mathrm{Fix}(T) \neq \emptyset$, is called quasi-nonexpansive, if

$$\|T(f) - h\| \leq \|f - h\|, \quad \forall f \in \mathcal{H}, \forall h \in \mathrm{Fix}(T).$$

The fixed point set of T is convex.



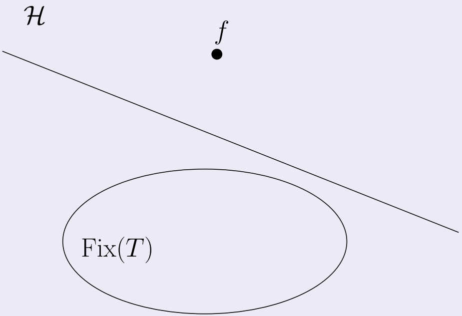
Quasi-nonexpansive Mapping

Definition (Quasi-nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$, with $\mathrm{Fix}(T) \neq \emptyset$, is called quasi-nonexpansive, if

$$\|T(f) - h\| \ \leq \|f - h\|, \quad \forall f \in \mathcal{H}, \forall h \in \mathrm{Fix}(T).$$

The fixed point set of T is convex.



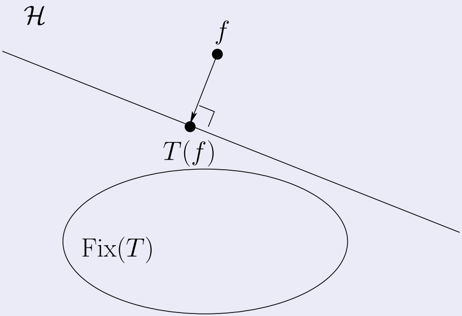
Quasi-nonexpansive Mapping

Definition (Quasi-nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$, with $\mathrm{Fix}(T) \neq \emptyset$, is called quasi-nonexpansive, if

$$\|T(f) - h\| \leq \|f - h\|, \quad \forall f \in \mathcal{H}, \forall h \in \mathrm{Fix}(T).$$

The fixed point set of T is convex.



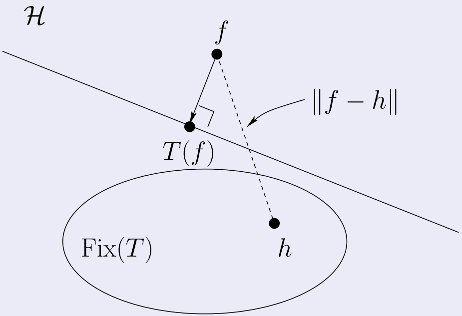
Quasi-nonexpansive Mapping

Definition (Quasi-nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$, with $\mathrm{Fix}(T) \neq \emptyset$, is called quasi-nonexpansive, if

$$\|T(f) - h\| \leq \|f - h\|, \quad \forall f \in \mathcal{H}, \forall h \in \mathrm{Fix}(T).$$

The fixed point set of T is convex.



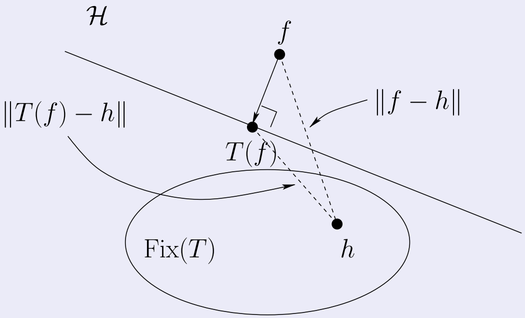
Quasi-nonexpansive Mapping

Definition (Quasi-nonexpansive Mapping)

A mapping $T : \mathcal{H} \to \mathcal{H}$, with $\mathrm{Fix}(T) \neq \emptyset$, is called quasi-nonexpansive, if

$$\|T(f) - h\| \leq \|f - h\|, \quad \forall f \in \mathcal{H}, \forall h \in \mathrm{Fix}(T).$$

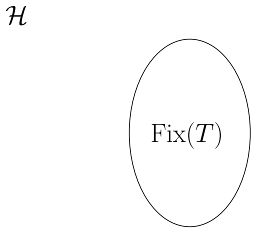
The fixed point set of T is convex.



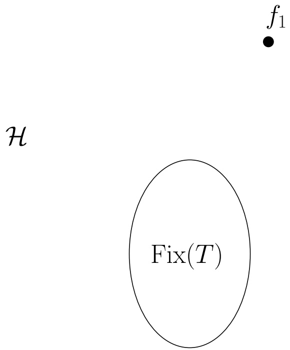
Every nonexpansive mapping is quasi-nonexpansive.

Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.

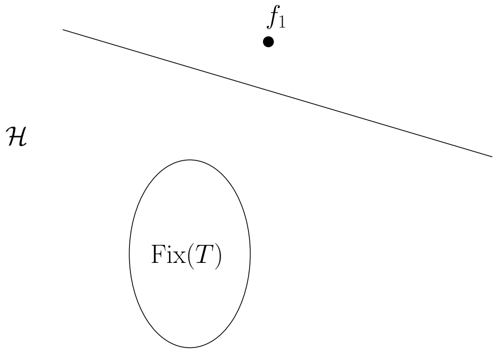
Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.



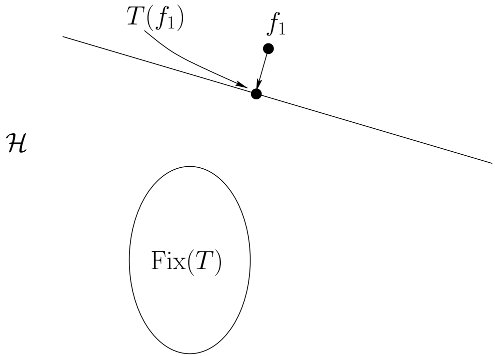
Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.



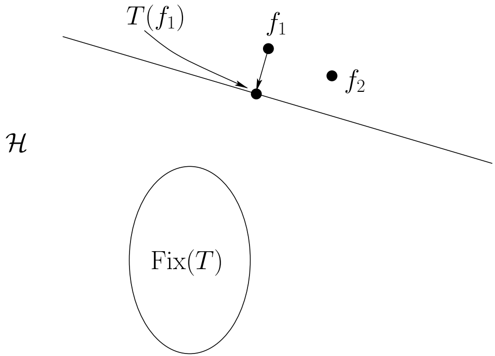
Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.



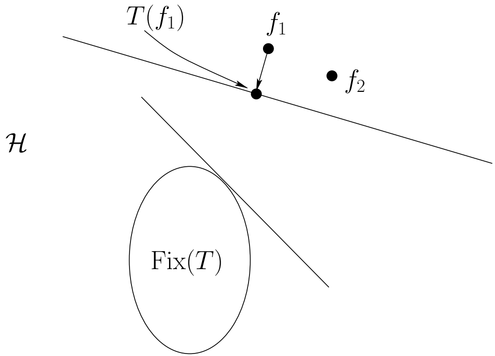
Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.



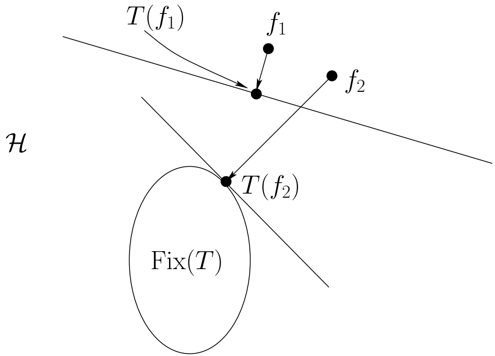
Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.



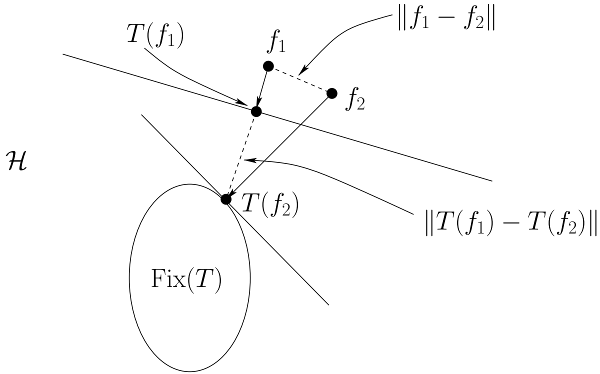
Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.



Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.



Projecting onto arbitrary separating hyperplanes generates a quasi-nonexpansive mapping which is not nonexpansive.



Definition (Subgradient)

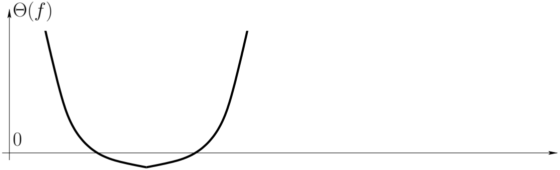
Given a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, the subgradient, $\Theta'(f)$, is an element of $\mathcal{H}$ such that

$$\langle g - f, \Theta'(f) \rangle + \Theta(f) \leq \Theta(g), \quad \forall g \in \mathcal{H}.$$

In other words, the hyperplane $\big\{\big(g, \langle g - f, \Theta'(f)\rangle + \Theta(f)\big) : g \in \mathcal{H}\big\}$, supports the graph of Θ at the point $\big(f, \Theta(f)\big)$.

Definition (Level set)

$$\operatorname{lev}_{\leq 0}(\Theta) := \{f \in \mathcal{H} : \Theta(f) \leq 0\}.$$



The Subgradient

Definition (Subgradient)

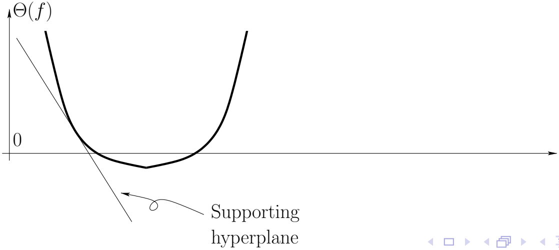
Given a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, the subgradient, $\Theta'(f)$, is an element of $\mathcal{H}$ such that

$$\langle g - f, \Theta'(f) \rangle + \Theta(f) \leq \Theta(g), \quad \forall g \in \mathcal{H}.$$

In other words, the hyperplane $\big\{\big(g, \langle g - f, \Theta'(f)\rangle + \Theta(f)\big) : g \in \mathcal{H}\big\}$, supports the graph of Θ at the point $\big(f, \Theta(f)\big)$.

Definition (Level set)

$$\text{lev}_{\leq 0}(\Theta) := \{f \in \mathcal{H} : \Theta(f) \leq 0\}.$$



The Subgradient

Definition (Subgradient)

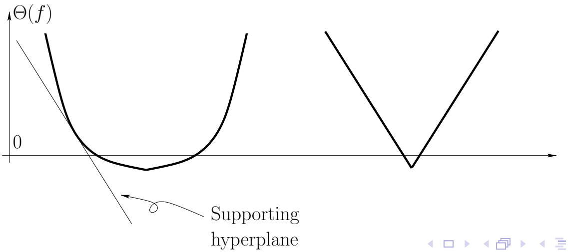
Given a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, the subgradient, $\Theta'(f)$, is an element of $\mathcal{H}$ such that

$$\langle g - f, \Theta'(f) \rangle + \Theta(f) \leq \Theta(g), \quad \forall g \in \mathcal{H}.$$

In other words, the hyperplane $\left\{ \left(g, \langle g - f, \Theta'(f) \rangle + \Theta(f) \right) : g \in \mathcal{H} \right\}$, supports the graph of Θ at the point $\left(f, \Theta(f) \right)$.

Definition (Level set)

$$\mathrm{lev}_{\leq 0}(\Theta) := \{ f \in \mathcal{H} : \Theta(f) \leq 0 \}.$$



The Subgradient

Definition (Subgradient)

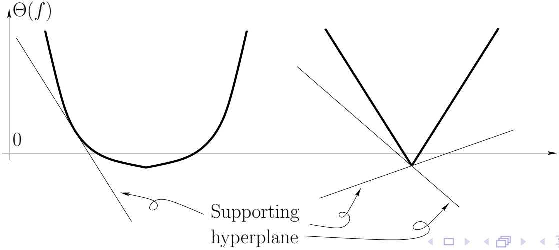
Given a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, the subgradient, $\Theta'(f)$, is an element of $\mathcal{H}$ such that

$$\langle g - f, \Theta'(f) \rangle + \Theta(f) \leq \Theta(g), \quad \forall g \in \mathcal{H}.$$

In other words, the hyperplane $\left\{ \left(g, \langle g - f, \Theta'(f) \rangle + \Theta(f) \right) : g \in \mathcal{H} \right\}$, supports the graph of Θ at the point $\left(f, \Theta(f) \right)$.

Definition (Level set)

$$\mathrm{lev}_{\leq 0}(\Theta) := \{ f \in \mathcal{H} : \Theta(f) \leq 0 \}.$$



Definition (Subgradient)

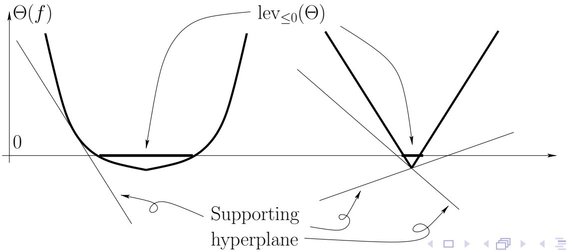
Given a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, the subgradient, $\Theta'(f)$, is an element of $\mathcal{H}$ such that

$$\langle g - f, \Theta'(f) \rangle + \Theta(f) \leq \Theta(g), \quad \forall g \in \mathcal{H}.$$

In other words, the hyperplane $\{(g, \langle g - f, \Theta'(f) \rangle + \Theta(f)) : g \in \mathcal{H}\}$, supports the graph of Θ at the point $(f, \Theta(f))$.

Definition (Level set)

$$\mathrm{lev}_{\leq 0}(\Theta) := \{f \in \mathcal{H} : \Theta(f) \leq 0\}.$$



The Subgradient Projection Mapping

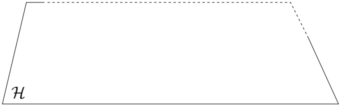
A Quasi-nonexpansive mapping

Definition (Subgradient projection mapping)

Let a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, with $\operatorname{lev}_{\leq 0}(\Theta) \neq \emptyset$. Then, the subgradient projection mapping $T_\Theta : \mathcal{H} \to \mathcal{H}$ is defined as follows:

$$T_\Theta(f) := \begin{cases} f - \frac{\Theta(f)}{\|\Theta'(f)\|^2}\Theta'(f), & \text{if } f \notin \operatorname{lev}_{\leq 0}(\Theta), \\ f, & \text{if } f \in \operatorname{lev}_{\leq 0}(\Theta). \end{cases}$$

The mapping T_Θ is a quasi-nonexpansive one.

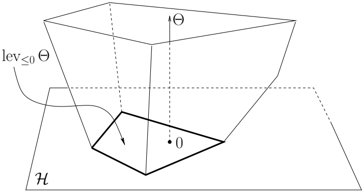


Definition (Subgradient projection mapping)

Let a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, with $\mathrm{lev}_{\leq 0}(\Theta) \neq \emptyset$. Then, the subgradient projection mapping $T_\Theta : \mathcal{H} \to \mathcal{H}$ is defined as follows:

$$T_\Theta(f) := \begin{cases} f - \frac{\Theta(f)}{\|\Theta'(f)\|^2}\Theta'(f), & \text{if } f \notin \mathrm{lev}_{\leq 0}(\Theta), \\ f, & \text{if } f \in \mathrm{lev}_{\leq 0}(\Theta). \end{cases}$$

The mapping T_Θ is a quasi-nonexpansive one.

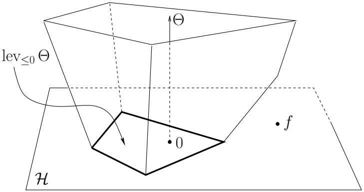


Definition (Subgradient projection mapping)

Let a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, with $\text{lev}_{\leq 0}(\Theta) \neq \emptyset$. Then, the subgradient projection mapping $T_\Theta : \mathcal{H} \to \mathcal{H}$ is defined as follows:

$$T_\Theta(f) := \begin{cases} f - \frac{\Theta(f)}{\|\Theta'(f)\|^2}\Theta'(f), & \text{if } f \notin \text{lev}_{\leq 0}(\Theta), \\ f, & \text{if } f \in \text{lev}_{\leq 0}(\Theta). \end{cases}$$

The mapping T_Θ is a quasi-nonexpansive one.

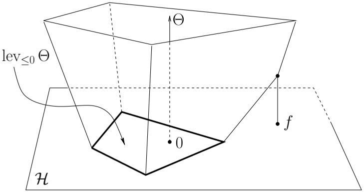


Definition (Subgradient projection mapping)

Let a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, with $\operatorname{lev}_{\leq 0}(\Theta) \neq \emptyset$. Then, the subgradient projection mapping $T_\Theta : \mathcal{H} \to \mathcal{H}$ is defined as follows:

$$T_\Theta(f) := \begin{cases} f - \frac{\Theta(f)}{\|\Theta'(f)\|^2}\Theta'(f), & \text{if } f \notin \operatorname{lev}_{\leq 0}(\Theta), \\ f, & \text{if } f \in \operatorname{lev}_{\leq 0}(\Theta). \end{cases}$$

The mapping T_Θ is a quasi-nonexpansive one.

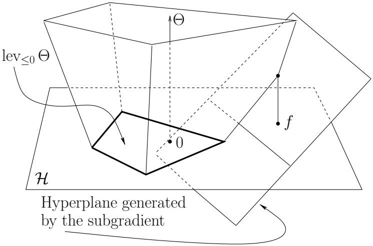


Definition (Subgradient projection mapping)

Let a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, with $\mathrm{lev}_{\leq 0}(\Theta) \neq \emptyset$. Then, the subgradient projection mapping $T_\Theta : \mathcal{H} \to \mathcal{H}$ is defined as follows:

$$T_\Theta(f) := \begin{cases} f - \frac{\Theta(f)}{\|\Theta'(f)\|^2}\Theta'(f), & \text{if } f \notin \mathrm{lev}_{\leq 0}(\Theta), \\ f, & \text{if } f \in \mathrm{lev}_{\leq 0}(\Theta). \end{cases}$$

The mapping T_Θ is a quasi-nonexpansive one.

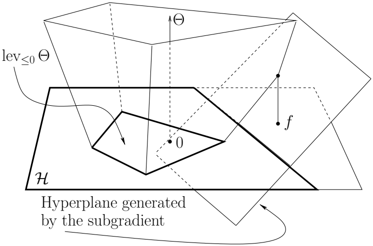


Definition (Subgradient projection mapping)

Let a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, with $\operatorname{lev}_{\leq 0}(\Theta) \neq \emptyset$. Then, the subgradient projection mapping $T_\Theta : \mathcal{H} \to \mathcal{H}$ is defined as follows:

$$T_\Theta(f) := \begin{cases} f - \frac{\Theta(f)}{\|\Theta'(f)\|^2}\Theta'(f), & \text{if } f \notin \operatorname{lev}_{\leq 0}(\Theta), \\ f, & \text{if } f \in \operatorname{lev}_{\leq 0}(\Theta). \end{cases}$$

The mapping T_Θ is a quasi-nonexpansive one.

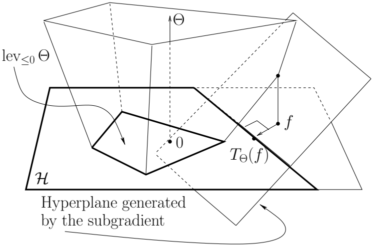


Definition (Subgradient projection mapping)

Let a convex function $\Theta : \mathcal{H} \to \mathbb{R}$, with $\mathrm{lev}_{\leq 0}(\Theta) \neq \emptyset$. Then, the subgradient projection mapping $T_\Theta : \mathcal{H} \to \mathcal{H}$ is defined as follows:

$$T_\Theta(f) := \begin{cases} f - \frac{\Theta(f)}{\|\Theta'(f)\|^2}\Theta'(f), & \text{if } f \notin \mathrm{lev}_{\leq 0}(\Theta), \\ f, & \text{if } f \in \mathrm{lev}_{\leq 0}(\Theta). \end{cases}$$

The mapping T_Θ is a quasi-nonexpansive one.



Generalizing the Previous Methodologies

Adaptive Projected Subgradient Method (APSM)

- For some user-defined
  - $\alpha \in (0, 1)$, $\lambda \in (0, 2)$,

- For some user-defined
  - $\alpha \in (0, 1)$, $\lambda \in (0, 2)$,
- and any initial point $f_0 \in \mathcal{H}$,

- For some user-defined
  - $\alpha \in (0,1)$, $\lambda \in (0,2)$,
- and any initial point $f_0 \in \mathcal{H}$,

APSM

- For some user-defined
  - $\alpha \in (0,1)$, $\lambda \in (0,2)$,
- and any initial point $f_0 \in \mathcal{H}$,

APSM

(f_n)

- For some user-defined
  - $\alpha \in (0,1)$, $\lambda \in (0,2)$,
- and any initial point $f_0 \in \mathcal{H}$,

APSM

$$\overbrace{\underbrace{\left((1-\lambda)I + \lambda T_{\Theta_n}\right)}_{=\text{ relaxed subgradient projection } T_{\Theta_n}}}^{\text{loss function info}}(f_n)$$

- For some user-defined
  - $\alpha \in (0, 1)$, $\lambda \in (0, 2)$,
- and any initial point $f_0 \in \mathcal{H}$,

APSM

$$f_{n+1} := \overbrace{\underbrace{\left(\alpha R_n + (1-\alpha)I\right)}_{=T_n:\ \text{averaged quasi-nonexpansive mapping}}}^{\text{a-priori info}} \overbrace{\underbrace{\left((1-\lambda)I + \lambda T_{\Theta_n}\right)}_{=\ \text{relaxed subgradient projection } T_{\Theta_n}}}^{\text{loss function info}} (f_n)$$

- For some user-defined
  - $\alpha \in (0,1)$, $\lambda \in (0,2)$,
- and any initial point $f_0 \in \mathcal{H}$,

APSM

$$f_{n+1} := \overbrace{\underbrace{\left(\alpha R_n + (1-\alpha) I\right)}_{=T_n:\ \text{averaged quasi-nonexpansive mapping}}}^{\text{a-priori info}} \overbrace{\underbrace{\left((1-\lambda) I + \lambda T_{\Theta_n}\right)}_{=\ \text{relaxed subgradient projection}\ T_{\Theta_n}}}^{\text{loss function info}} (f_n), \quad \forall n,$$

- For some user-defined
  - $\alpha \in (0,1)$, $\lambda \in (0,2)$,
- and any initial point $f_0 \in \mathcal{H}$,

APSM

$$f_{n+1} := \overbrace{\underbrace{\left(\alpha R_n + (1-\alpha) I\right)}_{=T_n:\ \text{averaged quasi-nonexpansive mapping}}}^{\text{a-priori info}} \overbrace{\underbrace{\left((1-\lambda) I + \lambda T_{\Theta_n}\right)}_{=\ \text{relaxed subgradient projection } T_{\Theta_n}}}^{\text{loss function info}} (f_n), \quad \forall n,$$

where

- $(R_n)_{n=0,1,\ldots}$ is a sequence of quasi-nonexpansive mappings. This sequence of mappings comprises the a-priori information.

- For some user-defined
  - $\alpha \in (0, 1)$, $\lambda \in (0, 2)$,
- and any initial point $f_0 \in \mathcal{H}$,

APSM

$$f_{n+1} := \overbrace{\underbrace{\left(\alpha R_n + (1-\alpha) I\right)}_{=T_n:\ \text{averaged quasi-nonexpansive mapping}}}^{\text{a-priori info}} \overbrace{\underbrace{\left((1-\lambda) I + \lambda T_{\Theta_n}\right)}_{=\ \text{relaxed subgradient projection } T_{\Theta_n}}}^{\text{loss function info}} (f_n), \quad \forall n,$$

where

- $(R_n)_{n=0,1,\ldots}$ is a sequence of quasi-nonexpansive mappings. This sequence of mappings comprises the a-priori information.
- $(\Theta_n)_{n=0,1,\ldots}$ is a sequence of loss/penalty function which quantifies the deviation of the sequential training data from the underlying model.

Candidates for the Loss Functions $(\Theta_n)_{n=0,1,\ldots}$

Given the current estimate f_n, define $\forall f \in \mathcal{H}$,

$$\Theta_n(f) := \begin{cases} \sum_{i=n-q+1}^{n} \frac{\omega_i^{(n)} d(f_n, S_i[\epsilon])}{\sum_{j=n-q+1}^{n} \omega_j^{(n)} d(f_n, S_j[\epsilon])} d(f, S_i[\epsilon]), & \text{if } f \notin \bigcap_{i=n-q+1}^{n} S_i[\epsilon], \\ 0, & \text{otherwise.} \end{cases}$$

Given the current estimate f_n, define $\forall f \in \mathcal{H}$,

$$\Theta_n(f) := \begin{cases} \sum_{i=n-q+1}^{n} \frac{\omega_i^{(n)} d(f_n, S_i[\epsilon])}{\sum_{j=n-q+1}^{n} \omega_j^{(n)} d(f_n, S_j[\epsilon])} d(f, S_i[\epsilon]), & \text{if } f \notin \bigcap_{i=n-q+1}^{n} S_i[\epsilon], \\ 0, & \text{otherwise.} \end{cases}$$

Then, the APSM becomes: $\forall n$,

Candidates for the Loss Functions $(\Theta_n)_{n=0,1,\ldots}$

Given the current estimate f_n, define $\forall f \in \mathcal{H}$,

$$\Theta_n(f) := \begin{cases} \sum_{i=n-q+1}^{n} \frac{\omega_i^{(n)} d(f_n, S_i[\epsilon])}{\sum_{j=n-q+1}^{n} \omega_j^{(n)} d(f_n, S_j[\epsilon])} d(f, S_i[\epsilon]), & \text{if } f \notin \bigcap_{i=n-q+1}^{n} S_i[\epsilon], \\ 0, & \text{otherwise.} \end{cases}$$

Then, the APSM becomes: $\forall n$,

$$\left(f_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} P_{S_i[\epsilon]}(f_n) - f_n \right) \right),$$

Candidates for the Loss Functions $(\Theta_n)_{n=0,1,\ldots}$

Given the current estimate f_n, define $\forall f \in \mathcal{H}$,

$$\Theta_n(f) := \begin{cases} \sum_{i=n-q+1}^{n} \frac{\omega_i^{(n)} d(f_n, S_i[\epsilon])}{\sum_{j=n-q+1}^{n} \omega_j^{(n)} d(f_n, S_j[\epsilon])} d(f, S_i[\epsilon]), & \text{if } f \notin \bigcap_{i=n-q+1}^{n} S_i[\epsilon], \\ 0, & \text{otherwise.} \end{cases}$$

Then, the APSM becomes: $\forall n$,

$$\underbrace{\left(\alpha R_n + (1-\alpha)I\right)}_{\substack{T_n:\ \text{averaged} \\ \text{quasi-nonexpansive mapping}}} \left(f_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} P_{S_i[\epsilon]}(f_n) - f_n \right) \right),$$

Candidates for the Loss Functions $(\Theta_n)_{n=0,1,\ldots}$

Given the current estimate f_n, define $\forall f \in \mathcal{H}$,

$$\Theta_n(f) := \begin{cases} \sum_{i=n-q+1}^{n} \frac{\omega_i^{(n)} d(f_n, S_i[\epsilon])}{\sum_{j=n-q+1}^{n} \omega_j^{(n)} d(f_n, S_j[\epsilon])} d(f, S_i[\epsilon]), & \text{if } f \notin \bigcap_{i=n-q+1}^{n} S_i[\epsilon], \\ 0, & \text{otherwise.} \end{cases}$$

Then, the APSM becomes: $\forall n$,

$$f_{n+1} = \underbrace{\left(\alpha R_n + (1-\alpha) I\right)}_{T_n:\ \text{averaged quasi-nonexpansive mapping}} \left(f_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} P_{S_i[\epsilon]}(f_n) - f_n \right) \right),$$

Candidates for the Loss Functions $(\Theta_n)_{n=0,1,\ldots}$

Given the current estimate f_n, define $\forall f \in \mathcal{H}$,

$$\Theta_n(f) := \begin{cases} \sum_{i=n-q+1}^{n} \frac{\omega_i^{(n)} d(f_n, S_i[\epsilon])}{\sum_{j=n-q+1}^{n} \omega_j^{(n)} d(f_n, S_j[\epsilon])} d(f, S_i[\epsilon]), & \text{if } f \notin \bigcap_{i=n-q+1}^{n} S_i[\epsilon], \\ 0, & \text{otherwise.} \end{cases}$$

Then, the APSM becomes: $\forall n$,

$$f_{n+1} = \underbrace{(\alpha R_n + (1-\alpha) I)}_{T_n:\ \text{averaged quasi-nonexpansive mapping}} \left(f_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} P_{S_i[\epsilon]}(f_n) - f_n \right) \right),$$

where the extrapolation coefficient $\mu_n \in (0, 2\mathcal{M}_n)$ with

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j=n-q+1}^{n} \omega_j^{(n)} \|P_{S_j[\epsilon]}(f_n) - f_n\|^2}{\|\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(f_n) - f_n\|^2}, & \text{if } \sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(f_n) \neq f_n, \\ 1, & \text{otherwise.} \end{cases}$$

Example (Examples of averaged quasi-nonexpansive mappings)

Example (Examples of averaged quasi-nonexpansive mappings)

- The projection P_C onto a closed convex set C of $\mathcal{H}$.

Candidates for the Averaged Quasi-nonexpansives T_n

Example (Examples of averaged quasi-nonexpansive mappings)

- The projection P_C onto a closed convex set C of $\mathcal{H}$.
  - The projection P_V onto an affine set V of $\mathbb{R}^m$, (beamforming).

Example (Examples of averaged quasi-nonexpansive mappings)

- The projection P_C onto a closed convex set C of $\mathcal{H}$.
  - The projection P_V onto an affine set V of $\mathbb{R}^m$, (beamforming).
  - The projection $P_{B_{\ell_1}[\delta]}$ onto the ℓ_1 ball, (sparsity-aware learning).

Example (Examples of averaged quasi-nonexpansive mappings)

- The projection P_C onto a closed convex set C of $\mathcal{H}$.
  - The projection P_V onto an affine set V of $\mathbb{R}^m$, (beamforming).
  - The projection $P_{B_{\ell_1}[\delta]}$ onto the ℓ_1 ball, (sparsity-aware learning).
  - The projections $(P_{B_{\ell_1}[\boldsymbol{w}_n,\delta]})_{n=0,1,\ldots}$ onto a sequence of weighted ℓ_1 balls, (sparsity-aware learning).

Candidates for the Averaged Quasi-nonexpansives T_n

Example (Examples of averaged quasi-nonexpansive mappings)

- The projection P_C onto a closed convex set C of $\mathcal{H}$.
  - The projection P_V onto an affine set V of $\mathbb{R}^m$, (beamforming).
  - The projection $P_{B_{\ell_1}[\delta]}$ onto the ℓ_1 ball, (sparsity-aware learning).
  - The projections $(P_{B_{\ell_1}[\boldsymbol{w}_n,\delta]})_{n=0,1,\ldots}$ onto a sequence of weighted ℓ_1 balls, (sparsity-aware learning).
- The composition of projections $P_{C_1}\cdots P_{C_p}$, where $C_1,\ldots C_p$ are closed convex sets with $\bigcap_{i=1}^{p} C_i \neq \emptyset$.

Example (Examples of averaged quasi-nonexpansive mappings)

- The projection P_C onto a closed convex set C of $\mathcal{H}$.
  - The projection P_V onto an affine set V of $\mathbb{R}^m$, (beamforming).
  - The projection $P_{B_{\ell_1}[\delta]}$ onto the ℓ_1 ball, (sparsity-aware learning).
  - The projections $(P_{B_{\ell_1}[\boldsymbol{w}_n,\delta]})_{n=0,1,\dots}$ onto a sequence of weighted ℓ_1 balls, (sparsity-aware learning).
- The composition of projections $P_{C_1}\cdots P_{C_p}$, where $C_1,\dots C_p$ are closed convex sets with $\bigcap_{i=1}^p C_i \neq \emptyset$.
  - The composition of projections $P_\Pi P_K$, where Π is a hyperplane and K is an icecream cone, (beamforming).

Example (Examples of averaged quasi-nonexpansive mappings)

- The projection P_C onto a closed convex set C of $\mathcal{H}$.
  - The projection P_V onto an affine set V of $\mathbb{R}^m$, (beamforming).
  - The projection $P_{B_{\ell_1}[\delta]}$ onto the ℓ_1 ball, (sparsity-aware learning).
  - The projections $(P_{B_{\ell_1}[\boldsymbol{w}_n,\delta]})_{n=0,1,\ldots}$ onto a sequence of weighted ℓ_1 balls, (sparsity-aware learning).
- The composition of projections $P_{C_1}\cdots P_{C_p}$, where $C_1,\ldots C_p$ are closed convex sets with $\bigcap_{i=1}^p C_i \neq \emptyset$.
  - The composition of projections $P_\Pi P_K$, where Π is a hyperplane and K is an icecream cone, (beamforming).
- The composition $P_{\mathcal{K}}\Big(I-\lambda\big(I-\sum_{i=1}^{p-1}\beta_i P_{C_i}\big)\Big)$, $\lambda\in(0,2)$, where $\mathcal{K}\cap\left(\bigcap_{i=1}^{p-1}C_i\right)=\emptyset$, (beamforming).

The Trip Still Goes on and the Topic Still Grows...

- Surprisingly, the APSM retains its performance and theoretical properties in the case where the Generalized Thresholding mapping T_{GT} is used in the place of T_n!

The Trip Still Goes on and the Topic Still Grows...

- Surprisingly, the APSM retains its performance and theoretical properties in the case where the Generalized Thresholding mapping T_{GT} is used in the place of T_n!
- Recall that $\mathrm{Fix}(T_{\text{GT}})$ is a union of subspaces, which is a non-convex set.

- Surprisingly, the APSM retains its performance and theoretical properties in the case where the Generalized Thresholding mapping T_{GT} is used in the place of T_n!
- Recall that $\mathrm{Fix}(T_{\mathrm{GT}})$ is a union of subspaces, which is a non-convex set.
- Such an application motivates the extension of the concept of a quasi-nonexpansive mapping to that of a partially quasi-nonexpansive one[10].

[10] [Kopsinis etal '11a].

Theoretical Properties

Define at $n \geq 0$, $\Omega_n := \mathrm{Fix}(T_n) \cap \mathrm{lev}_{\leq 0}\,\Theta_n$. Let $\Omega := \bigcap_{n \geq n_0} \Omega_n \neq \emptyset$, for some nonnegative integer n_0. Assume also that $\frac{\mu_n}{\mathcal{M}_n} \in [\epsilon_1, 2 - \epsilon_2]$, $\forall n \geq n_0$, for some sufficiently small $\epsilon_1, \epsilon_2 > 0$. Under the addition of some mild assumptions, the following statements hold true[11].

- Monotone approximation. $d(f_{n+1}, \Omega) \leq d(f_n, \Omega)$, $\forall n \geq n_0$.
- Asymptotic minimization. $\lim_{n\to\infty} \Theta_n(f_n) = 0$.
- Cluster points. If we assume that the set of all sequential strong cluster points $\mathfrak{S}\big((f_n)_{n=0,1,\ldots}\big)$ is nonempty, then

$$\mathfrak{S}\big((f_n)_{n=0,1,\ldots}\big) \subset \limsup_{n\to\infty} \mathrm{Fix}(T_n) \cap \limsup_{n\to\infty} \mathrm{lev}_{\leq 0}(\Theta_n),$$

  where $\limsup_{n\to\infty} A_n := \bigcap_{r>0} \bigcap_{n=1}^{\infty} \bigcup_{k=n}^{\infty} \big(A_k + B[0,r]\big)$, and $B[0,r]$ is a closed ball of center 0 and radius r.
- Strong convergence. Assume that there exists a hyperplane $\Pi \subset \mathcal{H}$ such that $\mathrm{ri}_{\Pi}(\Omega) \neq \emptyset$. Then, there exists an $f_* \in \mathcal{H}$ such that $\lim_{n\to\infty} f_n = f_*$.

[11] [Slavakis, Yamada, '11].

- K. Slavakis, P. Bouboulis, and S. Theodoridis. Adaptive multiregression in reproducing kernel Hilbert spaces: the multiaccess MIMO channel case. *IEEE Trans. Neural Networks and Learning Systems,* 23(2): 260–276, Feb. 2012,
- Y. Kopsinis, K. Slavakis, S. Theodoridis, and S. McLaughlin. Generalized thresholding sparsity-aware online learning in a union of subspaces, 2011, `http://arxiv.org/abs/1112.0665`.
- K. Slavakis and I. Yamada. The adaptive projected subgradient method constrained by families of quasi-nonexpansive mappings and its application to online learning, 2011, `http://arxiv.org/abs/1008.5231`.
- S. Theodoridis, K. Slavakis, and I. Yamada. Adaptive learning in a world of projections: a unifying framework for linear and nonlinear classification and regression tasks. *IEEE Signal Processing Magazine,* 28(1):97–123, 2011.
- Y. Kopsinis, K. Slavakis, and S. Theodoridis. Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls. *IEEE Trans. Signal Processing,* 59(3):936–952, 2011.
- S. Chouvardas, K. Slavakis, and S. Theodoridis. Adaptive robust distributed learning in diffusion sensor networks. *IEEE Trans. Signal Processing,* 59(10): 4692–4707, 2011.

Bibliography

- K. Slavakis, S. Theodoridis, and I. Yamada. Adaptive constrained learning in reproducing kernel Hilbert spaces: the robust beamforming case. *IEEE Trans. Signal Processing*, 57(12):4744–4764, Dec. 2009.
- K. Slavakis, S. Theodoridis, and I. Yamada. Online kernel-based classification using adaptive projection algorithms. *IEEE Trans. Signal Processing,* 56(7), Part 1: 2781–2796, July 2008.
- K. Slavakis and I. Yamada. Robust wideband beamforming by the hybrid steepest descent method. *IEEE Trans. Signal Processing,* 55(9): 4511–4522, Sept. 2007.
- K. Slavakis, I. Yamada, and N. Ogura. The adaptive projected subgradient method over the fixed point set of strongly attracting nonexpansive mappings. *Numerical Functional Analysis and Optimization*, 27(7&8):905–930, 2006.
- I. Yamada and N. Ogura. Adaptive projected subgradient method for asymptotic minimization of sequence of nonnegative convex functions. *Numerical Functional Analysis and Optimization*, 25(7&8):593–617, 2004.
- I. Yamada, K. Slavakis, and K. Yamada. An efficient robust adaptive filtering algorithm based on parallel subgradient projection techniques. *IEEE Trans. Signal Processing,* 50(5): 1091–1101, May 2002.

Matlab code

http://users.uop.gr/~slavakis/publications.htm