An Online Learning View via a Projections' Path in the Sparse-land

Sergios Theodoridis[1]

[1]Dept. of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece.

Workshop on Sparse Signal Processing
Friday, Sep. 16, 2016

Joint work with
P. Bouboulis, S. Chouvardas, Y. Kopsinis, G. Papageorgiou, K. Slavakis

Sparsity

Sparse Modeling

- Sparse modeling has been a major focus of research effort over the last decade or so.
- Sparsity promoting regularization of cost functions copes with:
  - Ill conditioning-overfitting when solving inverse problems; Learning from data is an instance of inverse problems.
  - **Promote** zeros when the underlying models have many near-to-zero values.

Sparsity

Sparse Modeling

- Sparse modeling has been a major focus of research effort over the last decade or so.
- Sparsity promoting regularization of cost functions copes with:
  - Ill conditioning-overfitting when solving inverse problems; Learning from data is an instance of inverse problems.
  - **Promote** zeros when the underlying models have many near-to-zero values.

Sparsity

Sparse Modeling

- Sparse modeling has been a major focus of research effort over the last decade or so.
- Sparsity promoting regularization of cost functions copes with:
  - Ill conditioning-overfitting when solving inverse problems; Learning from data is an instance of inverse problems.
  - **Promote** zeros when the underlying models have many near-to-zero values.

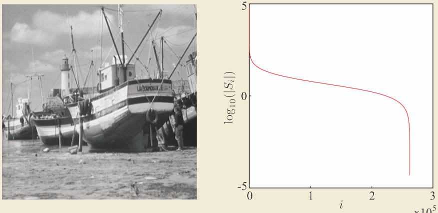
Sparsity

Sparse Modeling

- Sparse modeling has been a major focus of research effort over the last decade or so.
- Sparsity promoting regularization of cost functions copes with:
  - Ill conditioning-overfitting when solving inverse problems; Learning from data is an instance of inverse problems.
  - **Promote** zeros when the underlying models have many near-to-zero values.

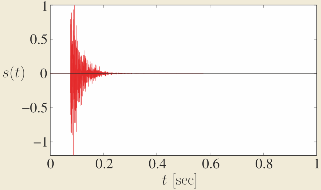
The need for sparse Models: Two examples

- Compression



(a) (b)

- Echo Cancelation



Sparse Modeling

The Generic Model

OUTPUT=INPUT $\times$ SPARSE MODEL+NOISE

Sparse Modeling

The Regression Model

- A generic model that covers a large class of problems (Filtering, Prediction)

$$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n$$

  - $\boldsymbol{a}_* \in \mathbb{R}^L$, is the unknown vector.
  - $\boldsymbol{u}_n \in \mathbb{R}^L$, is the incoming signal (sensing vectors).
  - $y_n \in \mathbb{R}$, is the observed signal (measurements).
  - v_n is the additive noise process.

- $\boldsymbol{a}_*$ is assumed to be sparse. That is, only a few, $K << L$, of its components are nonzero

$$\boldsymbol{a}_* = [0, 0, \underbrace{\star}_{1}, 0, \ldots, 0, \underbrace{\star}_{2}, 0, 0, \ldots, 0, \underbrace{\star}_{K}, 0, \ldots, 0]^T$$

- In its simplest formulation the task comprises the estimation of $\boldsymbol{a}_*$, based on a set of measurements $(y_n, \boldsymbol{u}_n)$, $n = 1 \ldots N$.

Sparse Modeling

The Regression Model

- A generic model that covers a large class of problems (Filtering, Prediction)

$$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n$$

  - $\boldsymbol{a}_* \in \mathbb{R}^L$, is the unknown vector.
  - $\boldsymbol{u}_n \in \mathbb{R}^L$, is the incoming signal (sensing vectors).
  - $y_n \in \mathbb{R}$, is the observed signal (measurements).
  - v_n is the additive noise process.

- $\boldsymbol{a}_*$ is assumed to be sparse. That is, only a few, $K << L$, of its components are nonzero

$$\boldsymbol{a}_* = [0, 0, \underbrace{\star}_{1}, 0, \ldots, 0, \underbrace{\star}_{2}, 0, 0, \ldots, 0, \underbrace{\star}_{K}, 0, \ldots, 0]^T$$

- In its simplest formulation the task comprises the estimation of $\boldsymbol{a}_*$, based on a set of measurements $(y_n, \boldsymbol{u}_n)$, $n = 1 \ldots N$.

Sparse Modeling

The Regression Model

- A generic model that covers a large class of problems (Filtering, Prediction)

$$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n$$

  - $\boldsymbol{a}_* \in \mathbb{R}^L$, is the unknown vector.
  - $\boldsymbol{u}_n \in \mathbb{R}^L$, is the incoming signal (sensing vectors).
  - $y_n \in \mathbb{R}$, is the observed signal (measurements).
  - v_n is the additive noise process.

- $\boldsymbol{a}_*$ is assumed to be sparse. That is, only a few, $K << L$, of its components are nonzero

$$\boldsymbol{a}_* = [0, 0, \underbrace{\star}_{1}, 0, \ldots, 0, \underbrace{\star}_{2}, 0, 0, \ldots, 0, \underbrace{\star}_{K}, 0, \ldots, 0]^T$$

- In its simplest formulation the task comprises the estimation of $\boldsymbol{a}_*$, based on a set of measurements $(y_n, \boldsymbol{u}_n)$, $n = 1 \ldots N$.

Sparse Modeling

The Regression Model

- A generic model that covers a large class of problems (Filtering, Prediction)

$$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n$$

  - $\boldsymbol{a}_* \in \mathbb{R}^L$, is the unknown vector.
  - $\boldsymbol{u}_n \in \mathbb{R}^L$, is the incoming signal (sensing vectors).
  - $y_n \in \mathbb{R}$, is the observed signal (measurements).
  - v_n is the additive noise process.

- $\boldsymbol{a}_*$ is assumed to be sparse. That is, only a few, $K << L$, of its components are nonzero

$$\boldsymbol{a}_* = [0, 0, \underbrace{\star}_{1}, 0, \ldots, 0, \underbrace{\star}_{2}, 0, 0, \ldots, 0, \underbrace{\star}_{K}, 0, \ldots, 0]^T$$

- In its simplest formulation the task comprises the estimation of $\boldsymbol{a}_*$, based on a set of measurements $(y_n, \boldsymbol{u}_n)$, $n = 1 \ldots N$.

Sparse Modeling

The Regression Model

- A generic model that covers a large class of problems (Filtering, Prediction)

$$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n$$

  - $\boldsymbol{a}_* \in \mathbb{R}^L$, is the unknown vector.
  - $\boldsymbol{u}_n \in \mathbb{R}^L$, is the incoming signal (sensing vectors).
  - $y_n \in \mathbb{R}$, is the observed signal (measurements).
  - v_n is the additive noise process.

- $\boldsymbol{a}_*$ is assumed to be sparse. That is, only a few, $K << L$, of its components are nonzero

$$\boldsymbol{a}_* = [0, 0, \underbrace{\star}_{1}, 0, \ldots, 0, \underbrace{\star}_{2}, 0, 0, \ldots, 0, \underbrace{\star}_{K}, 0, \ldots, 0]^T$$

- In its simplest formulation the task comprises the estimation of $\boldsymbol{a}_*$, based on a set of measurements $(y_n, \boldsymbol{u}_n)$, $n = 1 \ldots N$.

Sparse Modeling

The Regression Model

- A generic model that covers a large class of problems (Filtering, Prediction)

$$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n$$

  - $\boldsymbol{a}_* \in \mathbb{R}^L$, is the unknown vector.
  - $\boldsymbol{u}_n \in \mathbb{R}^L$, is the incoming signal (sensing vectors).
  - $y_n \in \mathbb{R}$, is the observed signal (measurements).
  - v_n is the additive noise process.

- $\boldsymbol{a}_*$ is assumed to be sparse. That is, only a few, $K << L$, of its components are nonzero

$$\boldsymbol{a}_* = [0, 0, \underbrace{\star}_{1}, 0, \ldots, 0, \underbrace{\star}_{2}, 0, 0, \ldots, 0, \underbrace{\star}_{K}, 0, \ldots, 0]^T$$

- In its simplest formulation the task comprises the estimation of $\boldsymbol{a}_*$, based on a set of measurements $(y_n, \boldsymbol{u}_n)$, $n = 1 \ldots N$.

Sparse Modeling

Dictionary Learning

- This is a powerful tool in analysing signals in terms of overcomplete basis vectors.

$$\underbrace{[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]}_{L \times N} = \underbrace{[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]}_{L \times m} \underbrace{[\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]}_{m \times N}, \quad m > L$$

$$Y = UA$$

- $\boldsymbol{y}_n, \in \mathbb{R}^L \ n = 1, 2, \ldots, N$, are the observation vectors.
- $\boldsymbol{u}_i \in \mathbb{R}^L, \ i = 1, 2, \ldots, m$, are the unknown atoms of the dictionary.
- $\boldsymbol{a}_n \in \mathbb{R}^m, \ n = 1, 2, \ldots, N$, are the vectors of the unknown weights, corresponding in the respective expansion of the nth input vector:
$$\boldsymbol{y}_n = \sum_{i=1}^{m} \boldsymbol{u}_i a_{ni}$$
- where, $\boldsymbol{a}_n, \ n = 1, 2, \ldots, N$, sparse vectors.

Sparse Modeling

Dictionary Learning

- This is a powerful tool in analysing signals in terms of overcomplete basis vectors.

$$\underbrace{[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]}_{L \times N} = \underbrace{[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]}_{L \times m} \underbrace{[\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]}_{m \times N}, \quad m > L$$

$$Y = UA$$

- $\boldsymbol{y}_n, \in \mathbb{R}^L$ $n = 1, 2, \ldots, N$, are the observation vectors.
- $\boldsymbol{u}_i \in \mathbb{R}^L$, $i = 1, 2, \ldots, m$, are the unknown atoms of the dictionary.
- $\boldsymbol{a}_n \in \mathbb{R}^m$, $n = 1, 2, \ldots, N$, are the vectors of the unknown weights, corresponding in the respective expansion of the nth input vector:

$$\boldsymbol{y}_n = \sum_{i=1}^{m} \boldsymbol{u}_i a_{ni}$$

- where, $\boldsymbol{a}_n$, $n = 1, 2, \ldots, N$, sparse vectors.

Sparse Modeling

Dictionary Learning

- This is a powerful tool in analysing signals in terms of overcomplete basis vectors.

$$\underbrace{[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]}_{L \times N} = \underbrace{[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m]}_{L \times m} \underbrace{[\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]}_{m \times N}, \quad m > L$$

$$Y = UA$$

- $\boldsymbol{y}_n, \in \mathbb{R}^L \ n = 1, 2, \ldots, N$, are the observation vectors.
- $\boldsymbol{u}_i \in \mathbb{R}^L, \ i = 1, 2, \ldots, m$, are the unknown atoms of the dictionary.
- $\boldsymbol{a}_n \in \mathbb{R}^m, \ n = 1, 2, \ldots, N$, are the vectors of the unknown weights, corresponding in the respective expansion of the nth input vector:

$$\boldsymbol{y}_n = \sum_{i=1}^{m} \boldsymbol{u}_i a_{ni}$$

- where, $\boldsymbol{a}_n, \ n = 1, 2, \ldots, N$, sparse vectors.

Sparse Modeling

Low Rank Matrix Factorization

- This task is at the heart of dimensionality reduction.

$$Y = UA = \sum_{i=1}^{r} \boldsymbol{u}_i \hat{\boldsymbol{a}}_i^T$$

$$\underbrace{[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]}_{L \times N} = \underbrace{[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r]}_{L \times r} \underbrace{\begin{bmatrix} \hat{\boldsymbol{a}}_1^T \\ \vdots \\ \hat{\boldsymbol{a}}_r{}^T \end{bmatrix}}_{r \times N}$$

- $r < N$.
- PCA performs low rank matrix factorization, by imposing sparsity on the singular values as well as orthogonality on U.

Sparse Modeling

Low Rank Matrix Factorization

- This task is at the heart of dimensionality reduction.

$$Y = UA = \sum_{i=1}^{r} \boldsymbol{u}_i \hat{\boldsymbol{a}}_i^T$$

$$\underbrace{[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]}_{L \times N} = \underbrace{[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r]}_{L \times r} \underbrace{\begin{bmatrix} \hat{\boldsymbol{a}}_1^T \\ \vdots \\ \hat{\boldsymbol{a}}_r{}^T \end{bmatrix}}_{r \times N}$$

- $r < N$.
- PCA performs low rank matrix factorization, by imposing sparsity on the singular values as well as orthogonality on U.

Sparse Modeling

Low Rank Matrix Factorization

- Matrix Completion is a special constrained version of low rank matrix factorization
- Y has missing elements and the lower rank matrix factorization is constrained to provide the non-missing elements at the respective positions

$$\hat{Y} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ * & * & * & * & * & * \end{bmatrix}$$

$$= \sum_{i=1}^{r} \boldsymbol{u}_i \hat{\boldsymbol{a}}_i^T$$

Sparse Modeling

Low Rank Matrix Factorization

- Matrix Completion is a special constrained version of low rank matrix factorization
- Y has missing elements and the lower rank matrix factorization is constrained to provide the non-missing elements at the respective positions

$$\hat{Y} = \begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ * & * & * & * & * & * \end{bmatrix}$$

$$= \sum_{i=1}^{r} \boldsymbol{u}_i \hat{\boldsymbol{a}}_i^T$$

Sparse Modeling

Low Rank Matrix Factorization

- Robust PCA is another special constrained version of low rank matrix factorization.

$$Y = L + V$$

L is a low rank matrix and V is a sparse matrix. The latter models OUTLIER NOISE. Being outlier is sparse.

- The goal of the task is to obtain estimates $\tilde{L}$ and $\tilde{V}$ by imposing sparsity on the singular values of Y as well as on the elements of V, constrained so that $Y = \tilde{L} + \tilde{V}$.

Low Rank Matrix Factorization

- Robust PCA is another special constrained version of low rank matrix factorization.

$$Y = L + V$$

L is a low rank matrix and V is a sparse matrix. The latter models OUTLIER NOISE. Being outlier is sparse.

- The goal of the task is to obtain estimates $\tilde{L}$ and $\tilde{V}$ by imposing sparsity on the singular values of Y as well as on the elements of V, constrained so that $Y = \tilde{L} + \tilde{V}$.

Sparse Modeling

Low Rank Matrix Factorization

- Robust PCA is another special constrained version of low rank matrix factorization.

$$Y = L + V$$

L is a low rank matrix and V is a sparse matrix. The latter models OUTLIER NOISE. Being outlier is sparse.

- The goal of the task is to obtain estimates $\tilde{L}$ and $\tilde{V}$ by imposing sparsity on the singular values of Y as well as on the elements of V, constrained so that $Y = \tilde{L} + \tilde{V}$.

Sparse Modeling

Robust Regression

- Robust Regression is an old problem, with a major impact coming from the works of Huber. The revival of interest is due to a new look via sparsity-aware learning techniques. For example, the noise may comprise a few large values (outliers) on top of the Gaussian component. Since the large values are only a few, they can be treated via sparse modeling arguments.

Sparse Regression Modeling

There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Sparse Regression Modeling

There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch Learning Problem

Linear Regression Model $\quad y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n$

- $\boldsymbol{U} := [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N]^T \in \mathbb{R}^{N \times L}$
- $\boldsymbol{y} := [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^N$, and $\boldsymbol{v} := [v_1, v_2, \ldots, v_N]^T \in \mathbb{R}^N$.

Batch Formulation: $\quad \boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$

Sparse Regression Modeling

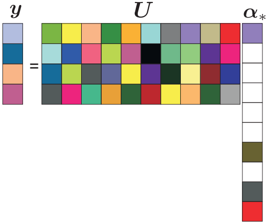
There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch Learning Problem

Linear Regression Model $\quad y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n$

- $\boldsymbol{U} := [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N]^T \in \mathbb{R}^{N \times L}$
- $\boldsymbol{y} := [y_1, y_2, \ldots, y_N]^T \in \mathbb{R}^N$, and $\boldsymbol{v} := [v_1, v_2, \ldots, v_N]^T \in \mathbb{R}^N$.

Batch Formulation: $\quad \boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$



Estimating the unknown

There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch vs Online Learning

Batch formulation: $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$

Online Formulation: $y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n,$

obtain an estimate, $\boldsymbol{a}_n$, after $(y_n, \boldsymbol{u}_n)$ has been received

Estimating the unknown

There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch vs Online Learning

Batch formulation: $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$

Online Formulation: $y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n,$

obtain an estimate, $\boldsymbol{a}_n$, after $(y_n, \boldsymbol{u}_n)$ has been received

Estimating the unknown

There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch vs Online Learning

Batch formulation: $$\boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$$

Online Formulation: $$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n,$$

obtain an estimate, $\boldsymbol{a}_n$, after $(y_n, \boldsymbol{u}_n)$ has been received

Estimating the unknown

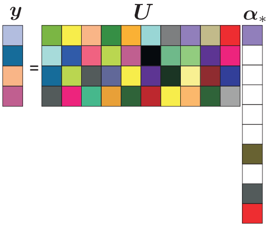
There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch vs Online Learning

Batch formulation: $$\boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$$

Online Formulation: $$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n,$$

obtain an estimate, $\boldsymbol{a}_n$, after $(y_n, \boldsymbol{u}_n)$ has been received



Estimating the unknown

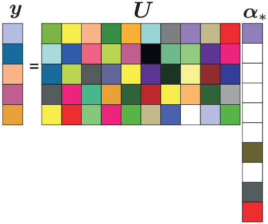
There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch vs Online Learning

Batch formulation: $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$

Online Formulation: $y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n,$

obtain an estimate, $\boldsymbol{a}_n$, after $(y_n, \boldsymbol{u}_n)$ has been received



Estimating the unknown

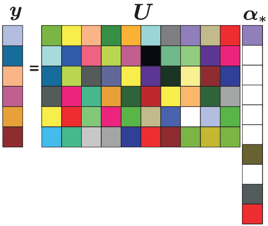
There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch vs Online Learning

Batch formulation: $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$

Online Formulation: $y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n,$

obtain an estimate, $\boldsymbol{a}_n$, after $(y_n, \boldsymbol{u}_n)$ has been received



Estimating the unknown

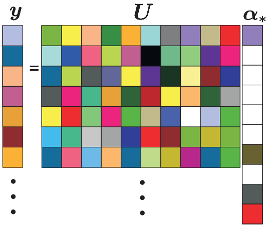
There are two paths that lead to the "truth", e.g, obtain an estimate $\hat{\boldsymbol{a}}$ of the unknown $\boldsymbol{a}_*$.

Batch vs Online Learning

Batch formulation: $$\boldsymbol{y} = \boldsymbol{U}\boldsymbol{a}_* + \boldsymbol{v}$$

Online Formulation: $$y_n = \boldsymbol{u}_n^T \boldsymbol{a}_* + v_n,$$

obtain an estimate, $\boldsymbol{a}_n$, after $(y_n, \boldsymbol{u}_n)$ has been received



Sparse Vs Online Learning

Sparsity-promoting Batch algorithm (Compressed Sensing)

- Are mobilized after a finite number of data, $(\boldsymbol{u}_n, y_n)_{n=0}^{N-1}$, is collected.
- For any new datum, the estimation of $\boldsymbol{a}_*$, is repeated from scratch.
- Computational complexity might become prohibitive.
- Excessive storage demands.
- It is a "mature" research field with a diverse number of techniques and applications.

Sparsity-promoting Online algorithms

- Infinite number of data.
- For any new datum, the estimate of $\boldsymbol{a}_*$ is updated dynamically.
- Cases of time-varying $\boldsymbol{a}_*$ are "naturally" handled.
- Low complexity is required for streaming applications.
- Fast convergence / Tracking.
- Large potential in Big Data applications

Sparse Vs Online Learning

Sparsity-promoting Batch algorithm (Compressed Sensing)

- Are mobilized after a finite number of data, $(\boldsymbol{u}_n, y_n)_{n=0}^{N-1}$, is collected.
- For any new datum, the estimation of $\boldsymbol{a}_*$, is repeated from scratch.
- Computational complexity might become prohibitive.
- Excessive storage demands.
- It is a "mature" research field with a diverse number of techniques and applications.

Sparsity-promoting Online algorithms

- Infinite number of data.
- For any new datum, the estimate of $\boldsymbol{a}_*$ is updated dynamically.
- Cases of time-varying $\boldsymbol{a}_*$ are "naturally" handled.
- Low complexity is required for streaming applications.
- Fast convergence / Tracking.
- Large potential in Big Data applications

Sparse Vs Online Learning

Sparsity-promoting Batch algorithm (Compressed Sensing)

- Are mobilized after a finite number of data, $(\boldsymbol{u}_n, y_n)_{n=0}^{N-1}$, is collected.
- For any new datum, the estimation of $\boldsymbol{a}_*$, is repeated from scratch.
- Computational complexity might become prohibitive.
- Excessive storage demands.
- It is a "mature" research field with a diverse number of techniques and applications.

Sparsity-promoting Online algorithms

- Infinite number of data.
- For any new datum, the estimate of $\boldsymbol{a}_*$ is updated dynamically.
- Cases of time-varying $\boldsymbol{a}_*$ are "naturally" handled.
- Low complexity is required for streaming applications.
- Fast convergence / Tracking.
- Large potential in Big Data applications

Sparsity-Promoting Methods

ℓ_0-norm constrained minimization

- ℓ_0 (pseudo) norm minimization: NP-hard nonconvex task.
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{a}\|_0, \quad \text{s.t.} \quad \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- The above is carried out via greedy-type algorithmic arguments.

Constrained Least Squares Estimation: Three equivalent formulations

- $\hat{\boldsymbol{a}} := \arg\min_{\boldsymbol{a}\in\mathbb{R}^l} \left\{\|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{a}\|_1\right\}$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2, \quad \text{s.t.} \quad \|\boldsymbol{a}\|_1 \leq \rho$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{a}\|_1, \quad \text{s.t.} \quad \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- Why ℓ_1 norm: It is the "closest" to ℓ_0 "norm" (number of nonzero elements) that retains its convex nature.

Sparsity-Promoting Methods

ℓ_0-norm constrained minimization

- ℓ_0 (pseudo) norm minimization: NP-hard nonconvex task.
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{a}\|_0,$ s.t. $\|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- The above is carried out via greedy-type algorithmic arguments.

Constrained Least Squares Estimation: Three equivalent formulations

- $\hat{\boldsymbol{a}} := \arg\min_{\boldsymbol{a}\in\mathbb{R}^l} \left\{\|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{a}\|_1\right\}$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2,$ s.t. $\|\boldsymbol{a}\|_1 \leq \rho$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{a}\|_1,$ s.t. $\|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- Why ℓ_1 norm: It is the "closest" to ℓ_0 "norm" (number of nonzero elements) that retains its convex nature.

Sparsity-Promoting Methods

ℓ_0-norm constrained minimization

- ℓ_0 (pseudo) norm minimization: NP-hard nonconvex task.
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a} \in \mathbb{R}^l} \|\boldsymbol{a}\|_0, \quad$ s.t. $\quad \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- The above is carried out via greedy-type algorithmic arguments.

Constrained Least Squares Estimation: Three equivalent formulations

- $\hat{\boldsymbol{a}} := \arg\min_{\boldsymbol{a} \in \mathbb{R}^l} \left\{ \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_1 \right\}$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a} \in \mathbb{R}^l} \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2, \quad$ s.t. $\quad \|\boldsymbol{a}\|_1 \leq \rho$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a} \in \mathbb{R}^l} \|\boldsymbol{a}\|_1, \quad$ s.t. $\quad \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- Why ℓ_1 norm: It is the "closest" to ℓ_0 "norm" (number of nonzero elements) that retains its convex nature.

Sparsity-Promoting Methods

ℓ_0-norm constrained minimization

- ℓ_0 (pseudo) norm minimization: NP-hard nonconvex task.
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{a}\|_0, \quad$ s.t. $\quad \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- The above is carried out via greedy-type algorithmic arguments.

Constrained Least Squares Estimation: Three equivalent formulations

- $\hat{\boldsymbol{a}} := \arg\min_{\boldsymbol{a}\in\mathbb{R}^l} \left\{\|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 + \lambda\|\boldsymbol{a}\|_1\right\}$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2, \quad$ s.t. $\quad \|\boldsymbol{a}\|_1 \leq \rho$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a}\in\mathbb{R}^l} \|\boldsymbol{a}\|_1, \quad$ s.t. $\quad \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- Why ℓ_1 norm: It is the "closest" to ℓ_0 "norm" (number of nonzero elements) that retains its convex nature.

Sparsity-Promoting Methods

ℓ_0-norm constrained minimization

- ℓ_0 (pseudo) norm minimization: NP-hard nonconvex task.
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a} \in \mathbb{R}^l} \|\boldsymbol{a}\|_0, \quad \text{s.t.} \quad \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- The above is carried out via greedy-type algorithmic arguments.

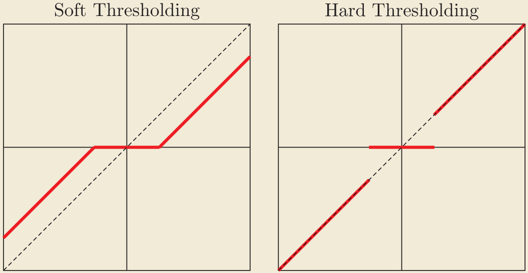
Constrained Least Squares Estimation: Three equivalent formulations

- $\hat{\boldsymbol{a}} := \arg\min_{\boldsymbol{a} \in \mathbb{R}^l} \left\{ \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 + \lambda \|\boldsymbol{a}\|_1 \right\}$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a} \in \mathbb{R}^l} \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2, \quad \text{s.t.} \quad \|\boldsymbol{a}\|_1 \leq \rho$
- $\hat{\boldsymbol{a}} : \min_{\boldsymbol{a} \in \mathbb{R}^l} \|\boldsymbol{a}\|_1, \quad \text{s.t.} \quad \|\boldsymbol{y} - U\boldsymbol{a}\|_2^2 \leq \epsilon$
- Why ℓ_1 norm: It is the "closest" to ℓ_0 "norm" (number of nonzero elements) that retains its convex nature.

Sparsity-Promoting Methods

Hard and Soft thresholding

- The ℓ_1 norm is associated with a soft thresholding operation on the respective coefficients. This is a continuous function operation, but it adds bias even for the large values. On the other hand, hard thresholding is a discontinuous one.



Batch Penalized Least-Squares Estimator

Penalized Least-Squares - General Case

$$\min_{\boldsymbol{a} \in \mathbb{R}^L} \left\{ \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{U}\boldsymbol{a}\|_2^2 + \lambda \sum_{i=1}^{L} p(|a_i|) \right\}$$

- $p(\cdot)$, sparsity-promoting penalty function,
- λ, regularization parameter.

Batch Penalized Least-Squares Estimator

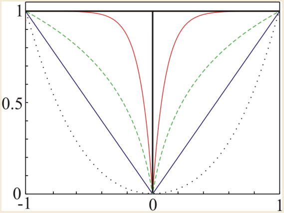
Penalized Least-Squares - General Case

$$\min_{\boldsymbol{a}\in\mathbb{R}^L}\left\{\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{U}\boldsymbol{a}\|_2^2+\lambda\sum_{i=1}^{L}p(|a_i|)\right\}$$

- $p(\cdot)$, sparsity-promoting penalty function,
- λ, regularization parameter.

Examples: Penalty functions

- $p(|a_i|) := |a_i|^\gamma, \ \forall a_i \in \mathbb{R}$
- $p(|a_i|) = \lambda\left(1 - e^{-\beta|a_i|}\right)$
- $p(|a_i|) := \frac{\lambda}{\log(\gamma+1)}\log(\gamma|a_i|+1), \ \forall a_i \in \mathbb{R}$



Online Sparsity-Promoting Methods

Penalized Recursive LS

$$\min_{\boldsymbol{a}\in\mathbb{R}^L} \left\{ \frac{1}{2}\sum_{n=1}^{N} \beta^{N-n} e_n^2 + \lambda \sum_{i=1}^{L} p(|a_i|) \right\},$$

Online Sparsity-Promoting Methods

Penalized Recursive LS

$$\min_{\boldsymbol{a}\in\mathbb{R}^L} \left\{ \frac{1}{2} \sum_{n=1}^{N} \beta^{N-n} e_n^2 + \lambda \sum_{i=1}^{L} p(|a_i|) \right\},$$

$$\boldsymbol{r}_N := \sum_{n=1}^{N} \beta^{N-n} y_n \boldsymbol{u}_n, \; \boldsymbol{R}_N := \sum_{n=1}^{N} \beta^{N-n} \boldsymbol{u}_n \boldsymbol{u}_n^T$$

Online Sparsity-Promoting Methods

Penalized Recursive LS

$$\min_{\boldsymbol{a}\in\mathbb{R}^L} \left\{ \frac{1}{2} \sum_{n=1}^{N} \beta^{N-n} e_n^2 + \lambda \sum_{i=1}^{L} p(|a_i|) \right\},$$

$$\boldsymbol{r}_{n+1} = \beta \boldsymbol{r}_n + y_{n+1} \boldsymbol{u}_{n+1}, \ \boldsymbol{R}_{n+1} = \beta \boldsymbol{R}_n + \boldsymbol{u}_{n+1} \boldsymbol{u}_{n+1}^T$$

Online Sparsity-Promoting Methods

Penalized Recursive LS

$$\min_{\boldsymbol{a}\in\mathbb{R}^L}\left\{\frac{1}{2}\sum_{n=1}^{N}\beta^{N-n}e_n^2+\lambda\sum_{i=1}^{L}p(|a_i|)\right\},$$

$$\boldsymbol{r}_{n+1}=\beta\boldsymbol{r}_n+y_{n+1}\boldsymbol{u}_{n+1},\ \ \boldsymbol{R}_{n+1}=\beta\boldsymbol{R}_n+\boldsymbol{u}_{n+1}\boldsymbol{u}_{n+1}^T$$

$$\boldsymbol{a}_{n+1}=f(\boldsymbol{r}_{n+1},\boldsymbol{R}_{n+1})$$

Online Sparsity-Promoting Methods

Penalized Recursive LS

$$\min_{\boldsymbol{a}\in\mathbb{R}^L} \left\{ \frac{1}{2} \sum_{n=1}^{N} \beta^{N-n} e_n^2 + \lambda \sum_{i=1}^{L} p(|a_i|) \right\},$$

$$\boldsymbol{r}_{n+1} = \beta \boldsymbol{r}_n + y_{n+1} \boldsymbol{u}_{n+1}, \;\; \boldsymbol{R}_{n+1} = \beta \boldsymbol{R}_n + \boldsymbol{u}_{n+1} \boldsymbol{u}_{n+1}^T$$

$$\boldsymbol{a}_{n+1} = f(\boldsymbol{r}_{n+1}, \boldsymbol{R}_{n+1})$$

- It Works!

- Complexity $\mathcal{O}(L^2)$
- Regularization parameter needs fine tuning

- [Angelosante, Bazerque and Giannakis, 2010]
- [Eksioglu and Tanc, 2011]

Online Sparsity-Promoting Methods

Penalized stochastic gradient descent: LMS type

$$\min_{\boldsymbol{a}\in\mathbb{R}^L} \left\{ \frac{1}{2} e_n^2 + \lambda \sum_{i=1}^{L} p(|a_i|) \right\}$$

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \mu e_n(\boldsymbol{a}) \boldsymbol{u}_n - \mu\lambda \boldsymbol{f}(\boldsymbol{a}_n)$$

$$\boldsymbol{f}(\boldsymbol{a}_n) = \left[\frac{\partial p(|a_{n,1}|)}{\partial a_{n,1}}, \frac{\partial p(|a_{n,2}|)}{\partial a_{n,2}}, \ldots, \frac{\partial p(|a_{n,L}|)}{\partial a_{n,L}} \right]^T$$

- Complexity $\mathcal{O}(L)$
- It Works! (when it is compared to standard LMS)
- Slow convergence
- Regularization parameter needs fine tuning

- [Chen, Gu and Hero, 2009]
- [Mileounis, Babadi, Kalouptsidis and Tarokh, 2010]
- [Wang and Gu, 2012]

Online Sparsity-Promoting Methods

Penalized stochastic gradient descent: LMS type

$$\min_{\boldsymbol{a}\in\mathbb{R}^L} \left\{ \frac{1}{2} e_n^2 + \lambda \sum_{i=1}^{L} p(|a_i|) \right\}$$

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \mu e_n(\boldsymbol{a}) \boldsymbol{u}_n - \mu\lambda \boldsymbol{f}(\boldsymbol{a}_n)$$

$$\boldsymbol{f}(\boldsymbol{a}_n) = \left[\frac{\partial p(|a_{n,1}|)}{\partial a_{n,1}}, \frac{\partial p(|a_{n,2}|)}{\partial a_{n,2}}, \ldots, \frac{\partial p(|a_{n,L}|)}{\partial a_{n,L}} \right]^T$$

- Complexity $\mathcal{O}(L)$
- It Works! (when it is compared to standard LMS)
- Slow convergence
- Regularization parameter needs fine tuning

- [Chen, Gu and Hero, 2009]
- [Mileounis, Babadi, Kalouptsidis and Tarokh, 2010]
- [Wang and Gu, 2012]

Online Sparsity-Promoting Methods

Penalized stochastic gradient descent: LMS type

$$\min_{\boldsymbol{a}\in\mathbb{R}^L} \left\{ \frac{1}{2} e_n^2 + \lambda \sum_{i=1}^{L} p(|a_i|) \right\}$$

$$\boldsymbol{a}_{n+1} = \boldsymbol{a}_n + \mu e_n(\boldsymbol{a}) \boldsymbol{u}_n - \mu\lambda \boldsymbol{f}(\boldsymbol{a}_n)$$

$$\boldsymbol{f}(\boldsymbol{a}_n) = \left[\frac{\partial p(|a_{n,1}|)}{\partial a_{n,1}}, \frac{\partial p(|a_{n,2}|)}{\partial a_{n,2}}, \ldots, \frac{\partial p(|a_{n,L}|)}{\partial a_{n,L}} \right]^T$$

- Complexity $\mathcal{O}(L)$
- It Works! (when it is compared to standard LMS)
- Slow convergence
- Regularization parameter needs fine tuning

- [Chen, Gu and Hero, 2009]
- [Mileounis, Babadi, Kalouptsidis and Tarokh, 2010]
- [Wang and Gu, 2012]

The Set-Theoretic Estimation Approach

The main concept

A descendent of POCS

The Set-Theoretic Estimation Approach

The main concept

A descendent of POCS

Projection onto a Closed Convex Set

Let C be a closed convex set in $\mathbb{R}^L$. Then, for each $\boldsymbol{a} \in \mathbb{R}^L$ there exists a unique $\boldsymbol{a}_* \in C$ such that

$$\|\boldsymbol{a} - \boldsymbol{a}_*\| = \min_{\boldsymbol{g} \in C} \|\boldsymbol{a} - \boldsymbol{g}\|.$$

The Set-Theoretic Estimation Approach

The main concept

A descendent of POCS

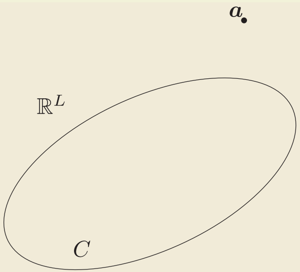
Projection onto a Closed Convex Set

Let C be a closed convex set in $\mathbb{R}^L$. Then, for each $\boldsymbol{a} \in \mathbb{R}^L$ there exists a unique $\boldsymbol{a}_* \in C$ such that

$$\|\boldsymbol{a} - \boldsymbol{a}_*\| = \min_{\boldsymbol{g} \in C} \|\boldsymbol{a} - \boldsymbol{g}\|.$$

Metric Projection Mapping

Metric Projection is the mapping
$P_C : \mathbb{R}^L \to C : \boldsymbol{a} \mapsto P_C(\boldsymbol{a}) := \boldsymbol{a}_*$.



The Set-Theoretic Estimation Approach

The main concept

A descendent of POCS

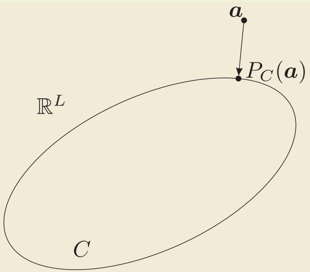
Projection onto a Closed Convex Set

Let C be a closed convex set in $\mathbb{R}^L$. Then, for each $\boldsymbol{a} \in \mathbb{R}^L$ there exists a unique $\boldsymbol{a}_* \in C$ such that

$$\|\boldsymbol{a} - \boldsymbol{a}_*\| = \min_{\boldsymbol{g} \in C} \|\boldsymbol{a} - \boldsymbol{g}\|.$$

Metric Projection Mapping

Metric Projection is the mapping
$P_C : \mathbb{R}^L \to C : \boldsymbol{a} \mapsto P_C(\boldsymbol{a}) := \boldsymbol{a}_*$.



The Set-Theoretic Estimation Approach

The main concept

A descendent of POCS

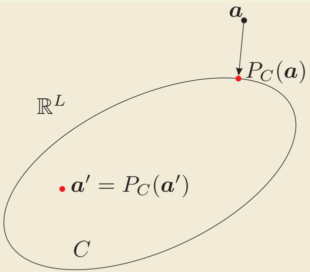
Projection onto a Closed Convex Set

Let C be a closed convex set in $\mathbb{R}^L$. Then, for each $\boldsymbol{a} \in \mathbb{R}^L$ there exists a unique $\boldsymbol{a}_* \in C$ such that

$$\|\boldsymbol{a} - \boldsymbol{a}_*\| = \min_{\boldsymbol{g} \in C} \|\boldsymbol{a} - \boldsymbol{g}\|.$$

Metric Projection Mapping

Metric Projection is the mapping
$P_C : \mathbb{R}^L \to C : \boldsymbol{a} \mapsto P_C(\boldsymbol{a}) := \boldsymbol{a}_*$.



The Set-Theoretic Estimation Approach

The main concept

A descendent of POCS

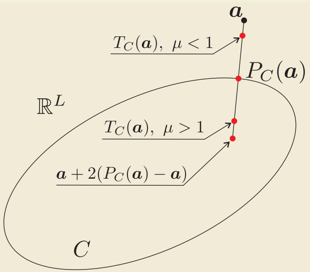
Projection onto a Closed Convex Set

Let C be a closed convex set in $\mathbb{R}^L$. Then, for each $\boldsymbol{a} \in \mathbb{R}^L$ there exists a unique $\boldsymbol{a}_* \in C$ such that

$$\|\boldsymbol{a} - \boldsymbol{a}_*\| = \min_{\boldsymbol{g} \in C} \|\boldsymbol{a} - \boldsymbol{g}\|.$$

Relaxed Projection Mapping

The relaxed Projection is the mapping
$T_C(\boldsymbol{a}) := \boldsymbol{a} + \mu(P_C(\boldsymbol{a}) - \boldsymbol{a})$,
$\mu \in (0, 2), \forall \boldsymbol{a} \in \mathbb{R}^L$.

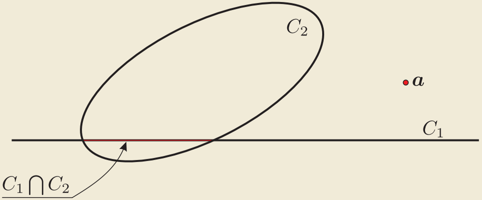


The Set-Theoretic Estimation Approach

The POCS: Finite number of Convex Sets [Von Neumann '33], [Bregman '65], [Gubin, Polyak, Raik '67]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^q C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. For any $\boldsymbol{a} \in \mathbb{R}^L$, define the sequence of projections:

$$P_{C_1}(\boldsymbol{a}).$$

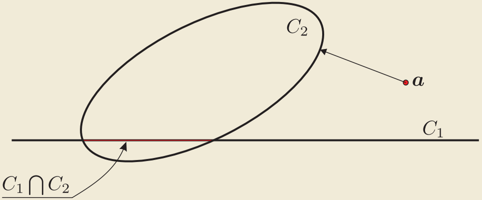


The Set-Theoretic Estimation Approach

The POCS: Finite number of Convex Sets [Von Neumann '33], [Bregman '65], [Gubin, Polyak, Raik '67]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^{q} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. For any $\boldsymbol{a} \in \mathbb{R}^L$, define the sequence of projections:

$$P_{C_1}(\boldsymbol{a}).$$

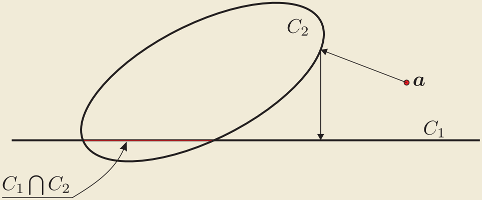


The Set-Theoretic Estimation Approach

The POCS: Finite number of Convex Sets [Von Neumann '33], [Bregman '65], [Gubin, Polyak, Raik '67]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^{q} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. For any $\boldsymbol{a} \in \mathbb{R}^L$, define the sequence of projections:

$$P_{C_2} P_{C_1}(\boldsymbol{a}).$$

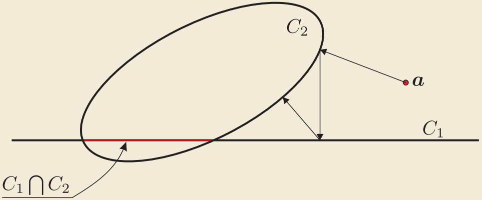


The Set-Theoretic Estimation Approach

The POCS: Finite number of Convex Sets [Von Neumann '33], [Bregman '65], [Gubin, Polyak, Raik '67]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^{q} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. For any $\boldsymbol{a} \in \mathbb{R}^L$, define the sequence of projections:

$$P_{C_1} P_{C_2} P_{C_1}(\boldsymbol{a}).$$

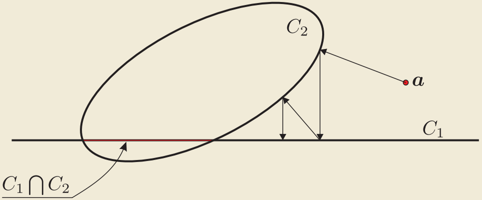


The Set-Theoretic Estimation Approach

The POCS: Finite number of Convex Sets [Von Neumann '33], [Bregman '65], [Gubin, Polyak, Raik '67]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^{q} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. For any $\boldsymbol{a} \in \mathbb{R}^L$, define the sequence of projections:

$$P_{C_2} P_{C_1} P_{C_2} P_{C_1}(\boldsymbol{a}).$$

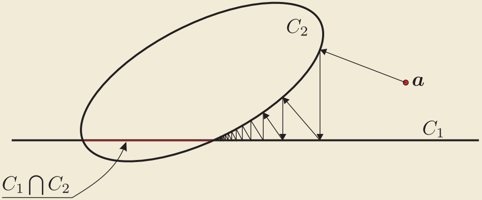


The Set-Theoretic Estimation Approach

The POCS: Finite number of Convex Sets [Von Neumann '33], [Bregman '65], [Gubin, Polyak, Raik '67]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^{q} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. For any $\boldsymbol{a} \in \mathbb{R}^L$, define the sequence of projections:

$$\cdots P_{C_2} P_{C_1} P_{C_2} P_{C_1}(\boldsymbol{a}).$$



The Set-Theoretic Estimation Approach

Convex Combination of Projection Mappings [Pierra '84]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^q C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. Let also a set of positive constants $w_1, \ldots, w_q$ such that $\sum_{i=1}^q w_i = 1$. Then for any $\boldsymbol{a}_0$, the sequence

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

converges weakly to a point $\boldsymbol{a}_*$ in $\bigcap_{i=1}^q C_i$,
where $\mu_n \in (\epsilon, \mathcal{M}_n)$, for $\epsilon \in (0,1)$, and
$\mathcal{M}_n := \frac{\sum_{i=1}^q w_i \|P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}{\|\sum_{i=1}^q w_i P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}$.

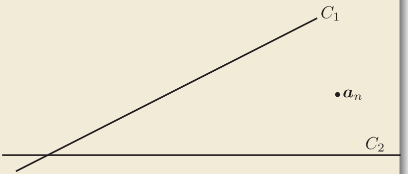
The Set-Theoretic Estimation Approach

Convex Combination of Projection Mappings [Pierra '84]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^q C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. Let also a set of positive constants $w_1, \ldots, w_q$ such that $\sum_{i=1}^q w_i = 1$. Then for any $\boldsymbol{a}_0$, the sequence

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

converges weakly to a point $\boldsymbol{a}_*$ in $\bigcap_{i=1}^q C_i$, where $\mu_n \in (\epsilon, \mathcal{M}_n)$, for $\epsilon \in (0,1)$, and $\mathcal{M}_n := \frac{\sum_{i=1}^q w_i \|P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}{\|\sum_{i=1}^q w_i P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}$.



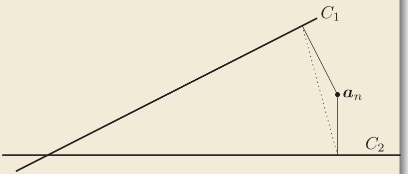
The Set-Theoretic Estimation Approach

Convex Combination of Projection Mappings [Pierra '84]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^q C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. Let also a set of positive constants $w_1, \ldots, w_q$ such that $\sum_{i=1}^q w_i = 1$. Then for any $\boldsymbol{a}_0$, the sequence

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

converges weakly to a point $\boldsymbol{a}_*$ in $\bigcap_{i=1}^q C_i$, where $\mu_n \in (\epsilon, \mathcal{M}_n)$, for $\epsilon \in (0,1)$, and $\mathcal{M}_n := \frac{\sum_{i=1}^q w_i \|P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}{\|\sum_{i=1}^q w_i P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}$.



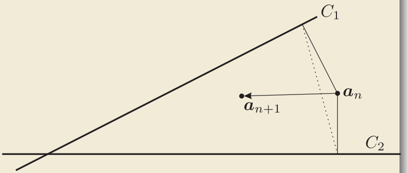
The Set-Theoretic Estimation Approach

Convex Combination of Projection Mappings [Pierra '84]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^q C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. Let also a set of positive constants $w_1, \ldots, w_q$ such that $\sum_{i=1}^q w_i = 1$. Then for any $\boldsymbol{a}_0$, the sequence

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

converges weakly to a point $\boldsymbol{a}_*$ in $\bigcap_{i=1}^q C_i$, where $\mu_n \in (\epsilon, \mathcal{M}_n)$, for $\epsilon \in (0,1)$, and $\mathcal{M}_n := \frac{\sum_{i=1}^q w_i \|P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}{\|\sum_{i=1}^q w_i P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}$.



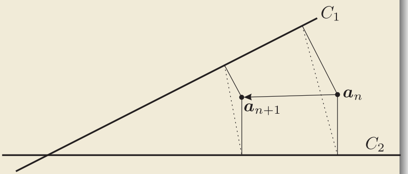
The Set-Theoretic Estimation Approach

Convex Combination of Projection Mappings [Pierra '84]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^q C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. Let also a set of positive constants $w_1, \ldots, w_q$ such that $\sum_{i=1}^q w_i = 1$. Then for any $\boldsymbol{a}_0$, the sequence

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

converges weakly to a point $\boldsymbol{a}_*$ in $\bigcap_{i=1}^q C_i$, where $\mu_n \in (\epsilon, \mathcal{M}_n)$, for $\epsilon \in (0,1)$, and $\mathcal{M}_n := \frac{\sum_{i=1}^q w_i \|P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}{\|\sum_{i=1}^q w_i P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}$.



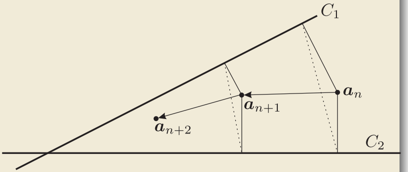
The Set-Theoretic Estimation Approach

Convex Combination of Projection Mappings [Pierra '84]

Given a finite number of closed convex sets $C_1, \ldots, C_q$, with $\bigcap_{i=1}^{q} C_i \neq \emptyset$, let their associated projection mappings be $P_{C_1}, \ldots, P_{C_q}$. Let also a set of positive constants $w_1, \ldots, w_q$ such that $\sum_{i=1}^{q} w_i = 1$. Then for any $\boldsymbol{a}_0$, the sequence

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

converges weakly to a point $\boldsymbol{a}_*$ in $\bigcap_{i=1}^{q} C_i$, where $\mu_n \in (\epsilon, \mathcal{M}_n)$, for $\epsilon \in (0,1)$, and $\mathcal{M}_n := \frac{\sum_{i=1}^{q} w_i \|P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}{\|\sum_{i=1}^{q} w_i P_{C_i}(\boldsymbol{a}_n) - \boldsymbol{a}_n\|^2}$.

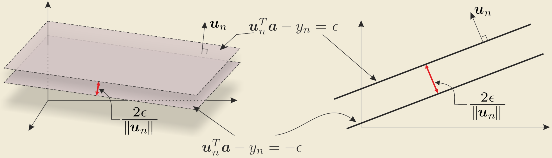


Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$S_n[\epsilon] :=$
$\{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\}$



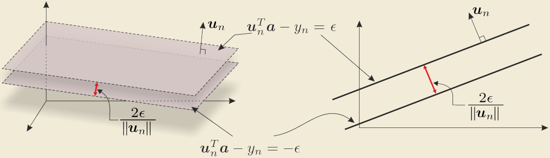
Solution

Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$S_n[\epsilon] :=$
$\{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\}$



Solution

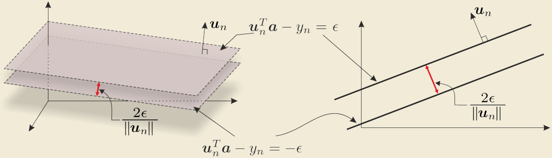
Find a point in the intersection of all the hyperslabs

Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$$S_n[\epsilon] := \{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\}$$



Solution

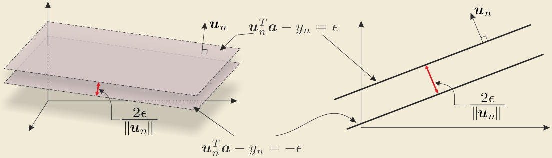
Find a point in the intersection of all the hyperslabs

Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

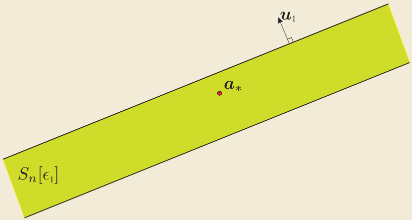
For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$$S_n[\epsilon] := \{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\}$$



Solution

Find a point in the intersection of all the hyperslabs

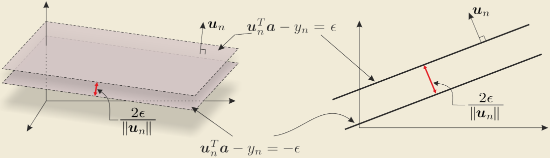


Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

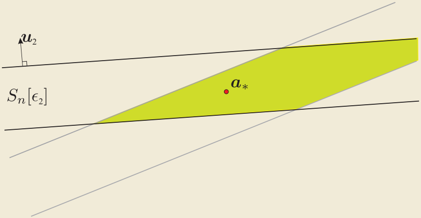
For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$$S_n[\epsilon] := \{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\}$$



Solution

Find a point in the intersection of all the hyperslabs

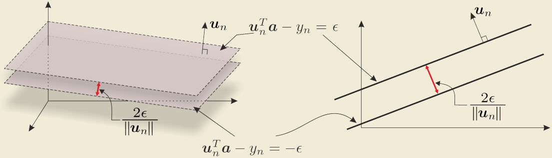


Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

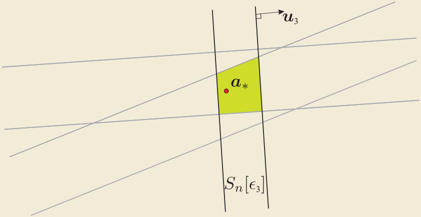
For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$$S_n[\epsilon] := \{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\}$$



Solution

Find a point in the intersection of all the hyperslabs

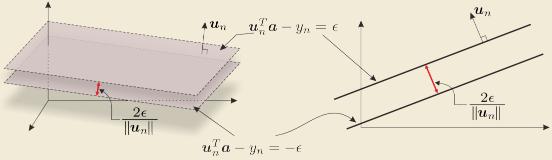


Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

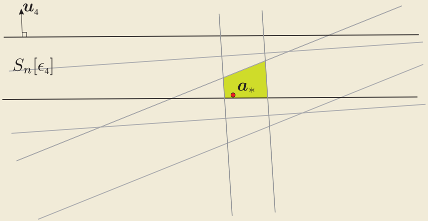
For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$$S_n[\epsilon] := \left\{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\right\}$$



Solution

Find a point in the intersection of all the hyperslabs

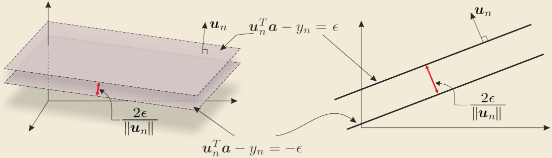


Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

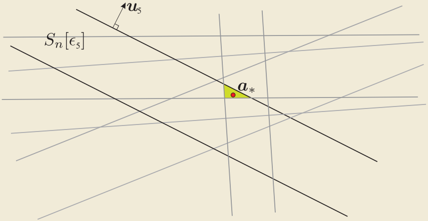
For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$$S_n[\epsilon] := \{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\}$$



Solution

Find a point in the intersection of all the hyperslabs

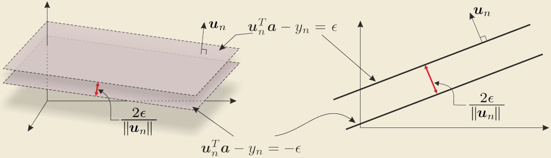


Set-Theoretic Estimation: The Online Case Approach

Constructing the Convex Sets

For each received set of measurements (training pairs) $(\boldsymbol{u}_n, y_n)$, construct a hyperslab:

$S_n[\epsilon] :=$
$\left\{\boldsymbol{a} \in \mathbb{R}^L : |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon\right\}$



Solution

[Yamada 2001], [Yamada, Slavakis, Yamada 2002], [Yamada, Ogura 2004], [Slavakis, Yamada Ogura 2006].

[Chouvardas, Slavakis, Theodoridis, Yamada, 2013]: Under the assumption of Bounded noise it converges with probability 1 arbitrarily close to the true model.

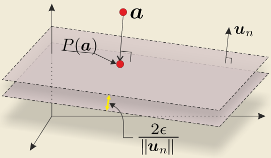
Adaptive Projection Subgradient Method (APSM)

The Algorithm

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

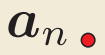
Projection onto Hyperslab

$$P_{S_n[\epsilon]}(\boldsymbol{a}) = \boldsymbol{a} + \begin{cases} \frac{y_n - \epsilon - \boldsymbol{u}_n^T \boldsymbol{a}}{\|\boldsymbol{u}_n\|^2} \boldsymbol{u}_n, & \text{if } y_n - \epsilon > \boldsymbol{u}_n^T \boldsymbol{a} \\ 0, & \text{if } |\boldsymbol{u}_n^T \boldsymbol{a} - y_n| \leq \epsilon \\ \frac{y_n + \epsilon - \boldsymbol{u}_n^T \boldsymbol{a}}{\|\boldsymbol{u}_n\|^2} \boldsymbol{u}_n, & \text{if } y_n + \epsilon < \boldsymbol{u}_n^T \boldsymbol{a} \end{cases}$$



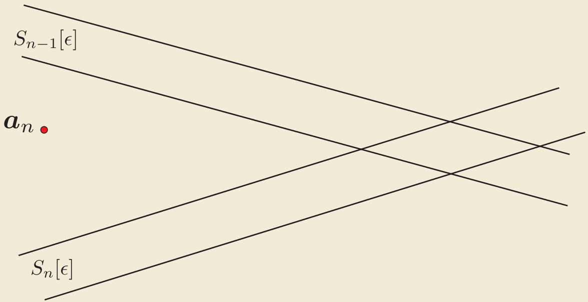
Adaptive Projection Subgradient Method (APSM)

Geometric illustration example



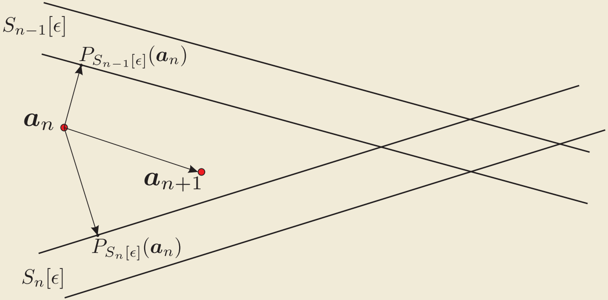
Adaptive Projection Subgradient Method (APSM)

Geometric illustration example



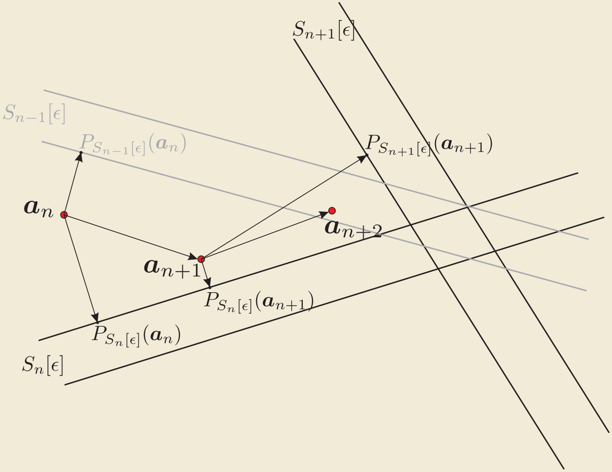
Adaptive Projection Subgradient Method (APSM)

Geometric illustration example



Adaptive Projection Subgradient Method (APSM)

Geometric illustration example



APSM under the ℓ_1 ball constraint

The ℓ_1-ball case

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

$$\left|\boldsymbol{a}^T \boldsymbol{u}_n - y_n\right| \leq \epsilon, \quad n = 0, 1, 2, \ldots$$
$$\|\boldsymbol{a}\|_1 \leq \delta.$$

- The recursion:

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

converges to

$$\boldsymbol{a}_* \in B_{\ell_1}[\delta] \cap \left(\bigcap_{n \geq n_0} S_n[\epsilon]\right).$$

APSM under the ℓ_1 ball constraint

The ℓ_1-ball case

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

$$\left|\boldsymbol{a}^T \boldsymbol{u}_n - y_n\right| \leq \epsilon, \quad n = 0, 1, 2, \ldots$$
$$\|\boldsymbol{a}\|_1 \leq \delta.$$

- The recursion:

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

converges to

$$\boldsymbol{a}_* \in B_{\ell_1}[\delta] \cap \left(\bigcap_{n \geq n_0} S_n[\epsilon]\right).$$

APSM under the ℓ_1 ball constraint

The ℓ_1-ball case

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

$$\left|\boldsymbol{a}^T \boldsymbol{u}_n - y_n\right| \leq \epsilon, \quad n = 0, 1, 2, \ldots$$
$$\|\boldsymbol{a}\|_1 \leq \delta.$$

- The recursion:

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

converges to

$$\boldsymbol{a}_* \in B_{\ell_1}[\delta] \cap \left(\bigcap_{n \geq n_0} S_n[\epsilon] \right).$$

APSM under the ℓ_1 ball constraint

The ℓ_1-ball case

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

$$\begin{aligned} &\left|\boldsymbol{a}^T \boldsymbol{u}_n - y_n\right| \leq \epsilon, \quad n = 0, 1, 2, \ldots \\ &\|\boldsymbol{a}\|_1 \leq \delta. \end{aligned}$$

- The recursion:

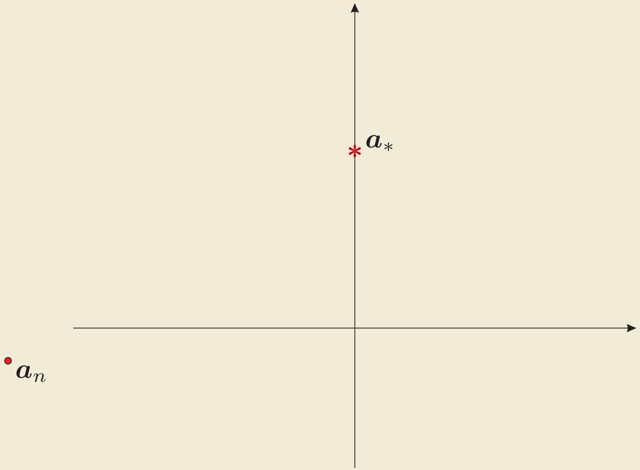
$$\boldsymbol{a}_{n+1} := P_{B_{\ell_1}[\delta]} \left(\boldsymbol{a}_n + \mu_n \left(\sum_{j=n-q+1}^{n} \omega_j^{(n)} P_{S_j[\epsilon]}(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right),$$

converges to

$$\boldsymbol{a}_* \in B_{\ell_1}[\delta] \cap \left(\bigcap_{n \geq n_0} S_n[\epsilon] \right).$$

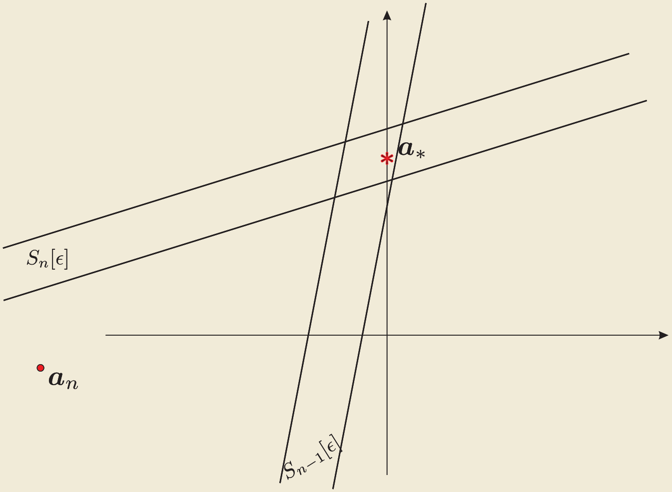
APSM under the ℓ_1 ball constraint

Geometric illustration example



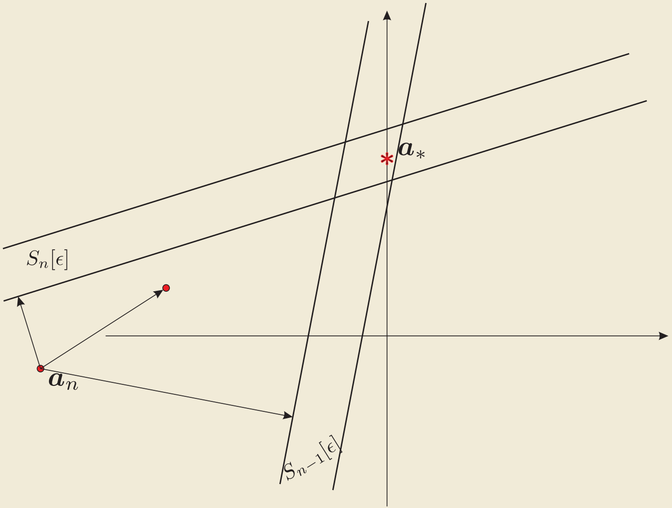
APSM under the ℓ_1 ball constraint

Geometric illustration example



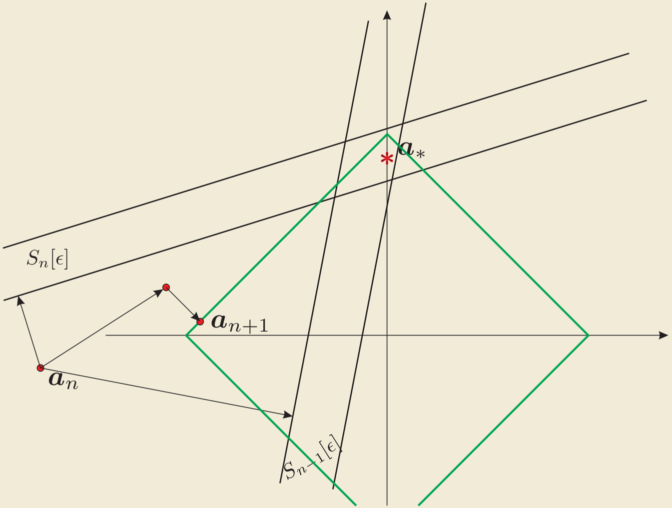
APSM under the ℓ_1 ball constraint

Geometric illustration example



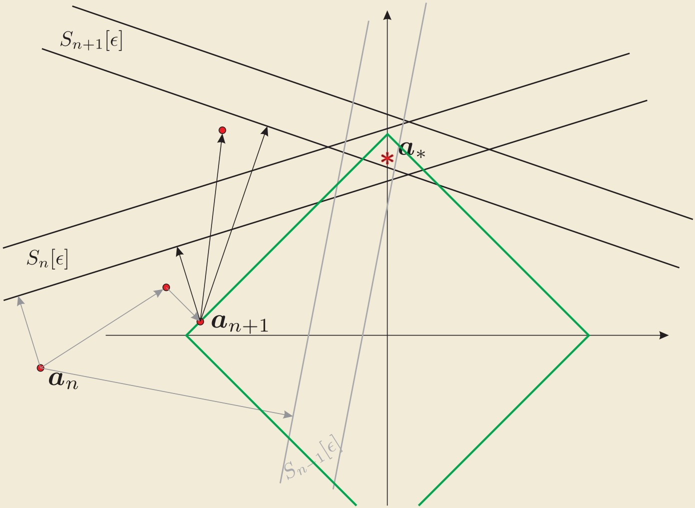
APSM under the ℓ_1 ball constraint

Geometric illustration example



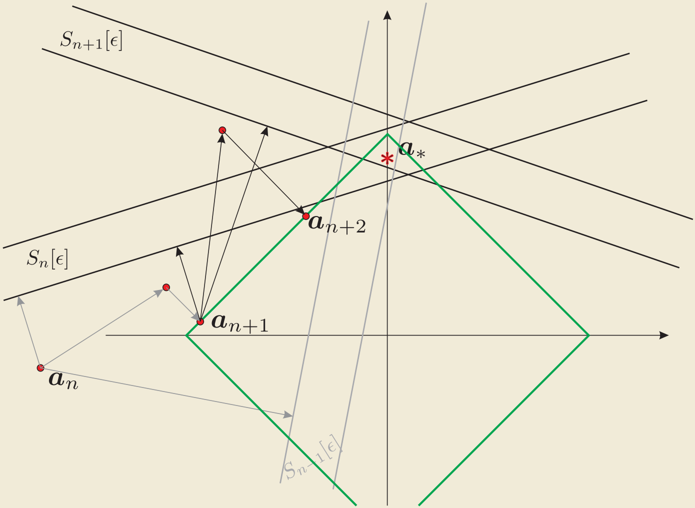
APSM under the ℓ_1 ball constraint

Geometric illustration example



APSM under the ℓ_1 ball constraint

Geometric illustration example



APSM under the weighted ℓ_1 ball constraint

The weighted ℓ_1-ball case:

- Convergence can be significantly speeded up if ℓ_1-ball, is replaced by the weighted ℓ_1 ball.
- Definition:

$$\|\boldsymbol{a}\|_{1,w} := \sum_{i=1}^{L} w_i |a_i|.$$

- Time-adaptive weighted norm:

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

- A time varying constraint case.
- The recursion:

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

APSM under the weighted ℓ_1 ball constraint

The weighted ℓ_1-ball case:

- Convergence can be significantly speeded up if ℓ_1-ball, is replaced by the weighted ℓ_1 ball.
- Definition:

$$\|\boldsymbol{a}\|_{1,w} := \sum_{i=1}^{L} w_i |a_i|.$$

- Time-adaptive weighted norm:

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

- A time varying constraint case.
- The recursion:

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

APSM under the weighted ℓ_1 ball constraint

The weighted ℓ_1-ball case:

- Convergence can be significantly speeded up if ℓ_1-ball, is replaced by the weighted ℓ_1 ball.
- Definition:

$$\|\boldsymbol{a}\|_{1,w} := \sum_{i=1}^{L} w_i |a_i|.$$

- Time-adaptive weighted norm:

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

- A time varying constraint case.
- The recursion:

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

APSM under the weighted ℓ_1 ball constraint

The weighted ℓ_1-ball case:

- Convergence can be significantly speeded up if ℓ_1-ball, is replaced by the weighted ℓ_1 ball.
- Definition:

$$\|\boldsymbol{a}\|_{1,w} := \sum_{i=1}^{L} w_i |a_i|.$$

- Time-adaptive weighted norm:

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

- A time varying constraint case.
- The recursion:

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

APSM under the weighted ℓ_1 ball constraint

The weighted ℓ_1-ball case:

- Convergence can be significantly speeded up if ℓ_1-ball, is replaced by the weighted ℓ_1 ball.
- Definition:

$$\|\boldsymbol{a}\|_{1,w} := \sum_{i=1}^{L} w_i |a_i|.$$

- Time-adaptive weighted norm:

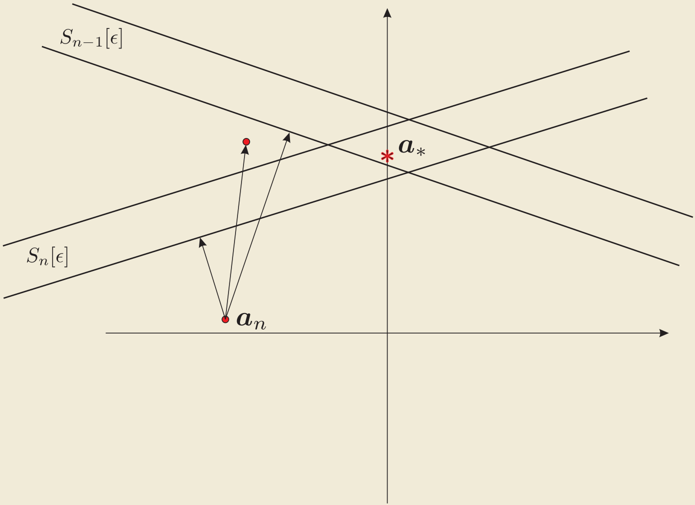
$$w_{n,i} := \frac{1}{|a_{n,i}| + \epsilon'_n}.$$

- A time varying constraint case.
- The recursion:

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

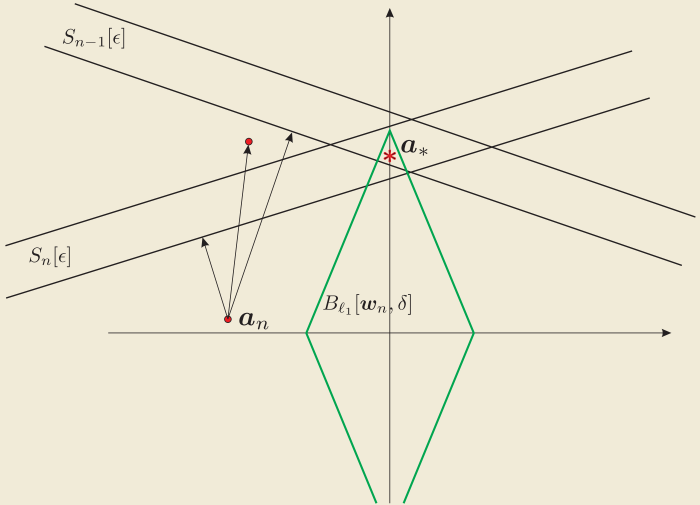
APSM under the weighted ℓ_1 ball constraint

Geometric illustration example



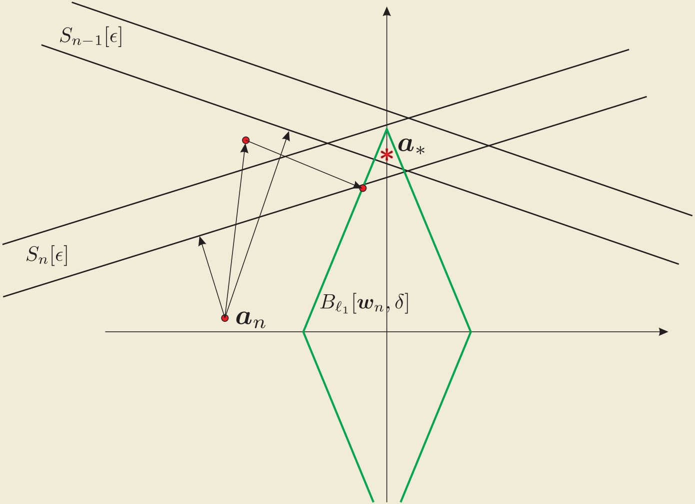
APSM under the weighted ℓ_1 ball constraint

Geometric illustration example



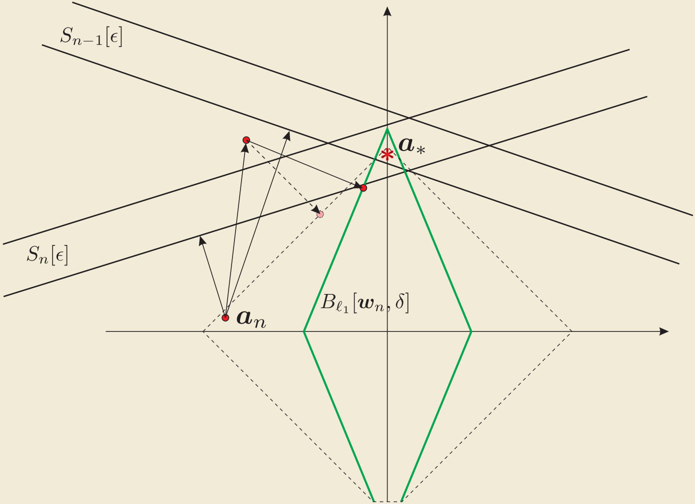
APSM under the weighted ℓ_1 ball constraint

Geometric illustration example



APSM under the weighted ℓ_1 ball constraint

Geometric illustration example



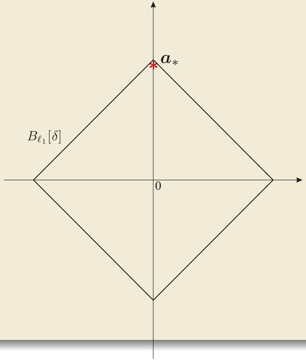
APSM under the weighted ℓ_1 ball constraint

Convergence of the Scheme

- Does this scheme converge?

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

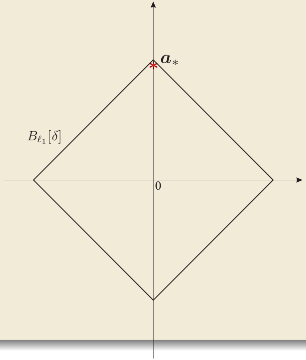
  Remark: This case was not covered by the existing theory.



APSM under the weighted ℓ_1 ball constraint

Convergence of the Scheme

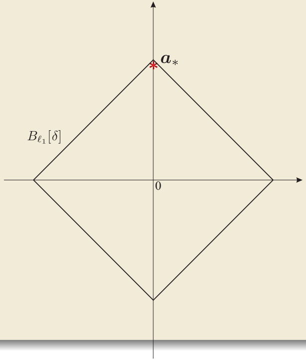
- Does this scheme converge?
  Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.
  Remark: This case was not covered by the existing theory.



APSM under the weighted ℓ_1 ball constraint

Convergence of the Scheme

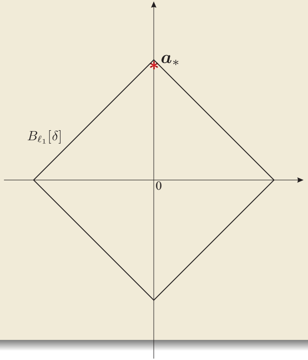
- Does this scheme converge?
  Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.
  Remark: This case was not covered by the existing theory.



APSM under the weighted ℓ_1 ball constraint

Convergence of the Scheme

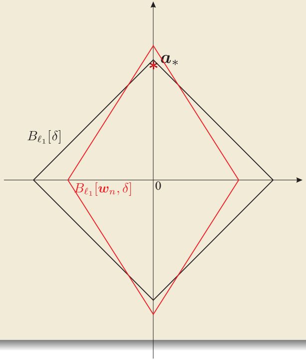
- Does this scheme converge?
  Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.
  Remark: This case was not covered by the existing theory.



APSM under the weighted ℓ_1 ball constraint

Convergence of the Scheme

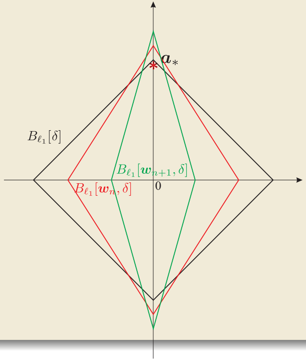
- Does this scheme converge?
  Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.
  Remark: This case was not covered by the existing theory.



APSM under the weighted ℓ_1 ball constraint

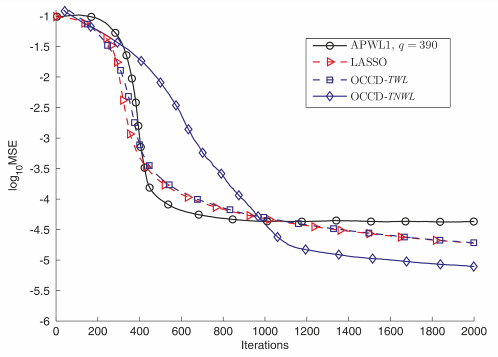
Convergence of the Scheme

- Does this scheme converge?
  Note that our constraint, i.e., the weighted ℓ_1-ball is a time-varying constraint.
  Remark: This case was not covered by the existing theory.



Simulation Examples

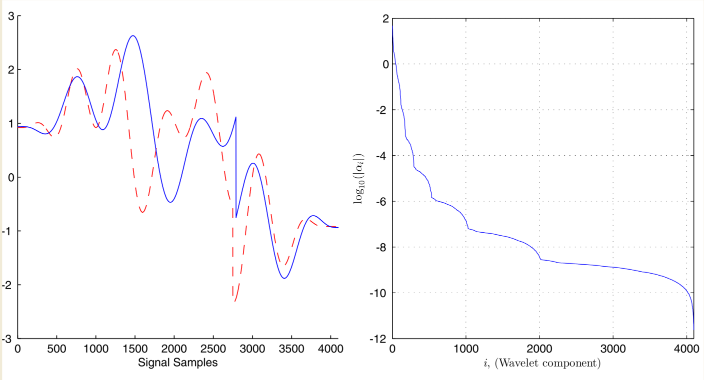
Example: Time-invariant signal sparse in wavelet domain



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

Simulation Examples

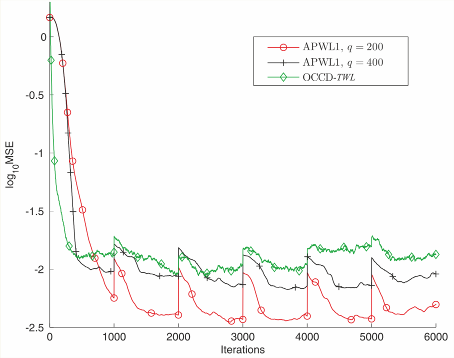
Example: Time varying signal compressible in wavelet domain



$L := 4096.$

The sum of two chirp signals.

Example: Time varying signal compressible in wavelet domain



$L := 4096$. The radius of the ℓ_1-ball is set to $\delta := 40$.

Movies of the OCCD, and the APWL1sub.

Generalized Thresholding Rules

Thresholding rules associated with non-convex penalty functions

- Penalized LS thresholding operators:

$$\min_{\boldsymbol{a}} \frac{1}{2}(\tilde{\boldsymbol{a}} - \boldsymbol{a})^2 + \lambda p(|\boldsymbol{a}|)$$

Generalized Thresholding Rules

Thresholding rules associated with non-convex penalty functions

- Penalized LS thresholding operators:

$$\min_{\boldsymbol{a}} \frac{1}{2}(\tilde{\boldsymbol{a}} - \boldsymbol{a})^2 + \lambda p(|\boldsymbol{a}|)$$

- $p(\cdot)$: nonnegative, nondecreasing and differentiable function on $(0, \infty)$

Generalized Thresholding Rules

Thresholding rules associated with non-convex penalty functions

- Penalized LS thresholding operators:

$$\min_{\boldsymbol{a}} \frac{1}{2}(\tilde{\boldsymbol{a}} - \boldsymbol{a})^2 + \lambda p(|\boldsymbol{a}|)$$

- $p(\cdot)$: nonnegative, nondecreasing and differentiable function on $(0, \infty)$
- Under some general conditions it has a unique solution [Antoniadis 2007].

Generalized Thresholding Rules

Thresholding rules associated with non-convex penalty functions

- Penalized LS thresholding operators:

$$\min_{\boldsymbol{a}} \frac{1}{2}(\tilde{\boldsymbol{a}} - \boldsymbol{a})^2 + \lambda p(|\boldsymbol{a}|)$$

- $p(\cdot)$: nonnegative, nondecreasing and differentiable function on $(0, \infty)$
- Under some general conditions it has a unique solution [Antoniadis 2007].
- PLSTO basically defines a mapping

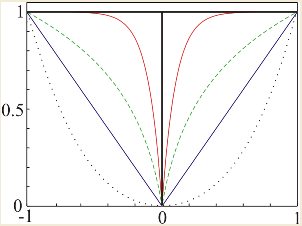
$$\tilde{\boldsymbol{a}} \mapsto \min_{\boldsymbol{a}} \frac{1}{2}(\tilde{\boldsymbol{a}} - \boldsymbol{a})^2 + \lambda p(|\boldsymbol{a}|)$$

  which corresponds to a Shrinkage operator.

Generalized Thresholding Rules

Examples: Penalty functions

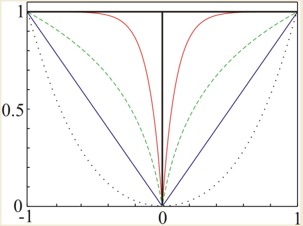
- $p(|a|) := |a|^\gamma, \ \forall a \in \mathbb{R}$
- $p(|a|) = \lambda \left(1 - e^{-\beta|a|}\right)$
- $p(|a|) := \frac{\lambda}{\log(\gamma+1)} \log(\gamma|a| + 1), \ \forall a \in \mathbb{R}$



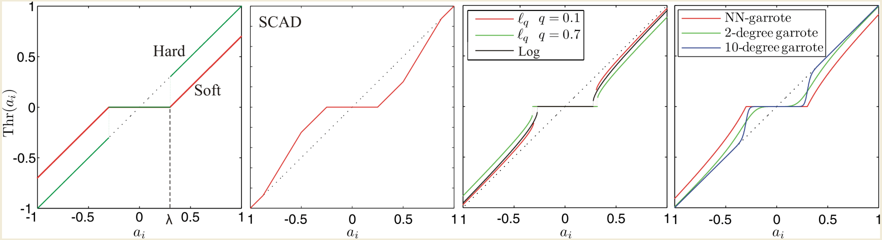
Generalized Thresholding Rules

Examples: Penalty functions

- $p(|a|) := |a|^{\gamma}, \ \forall a \in \mathbb{R}$
- $p(|a|) = \lambda \left(1 - e^{-\beta|a|}\right)$
- $p(|a|) := \frac{\lambda}{\log(\gamma+1)} \log(\gamma|a| + 1), \ \forall a \in \mathbb{R}$



Examples: Penalized Least-Squares Thresholding Operators



Generalized Thresholding Rules

Generalized Thresholding (GT) operator: Definition:

For any $\boldsymbol{a} \in \mathbb{R}^L$, $\boldsymbol{z} := T_{\mathsf{GT}}^{(K)}(\boldsymbol{a})$ is obtained coordinate-wise:

$$\forall l \in \overline{1, L}, \quad z_l := \begin{cases} a_l, & \text{If, } a_l \text{ is one of the largest } K \text{ components,} \\ \mathsf{shr}(a_l), & \text{otherwise} \end{cases}$$

Shrinkage Function (Shr)

- $\tau \mathsf{shr}(\tau) \geq 0$, $\forall \tau \in \mathbb{R}$.
- shr acts as a *strict* shrinkage operator over all intervals which do not include 0.
- Any arbitrary function inline with the properties above can be used.
- All the penalized Least-Squares thresholding operators are included.

Generalized Thresholding Rules

Generalized Thresholding **(GT)** operator: Definition:

For any $\boldsymbol{a} \in \mathbb{R}^L$, $\boldsymbol{z} := T_{\mathsf{GT}}^{(K)}(\boldsymbol{a})$ is obtained coordinate-wise:

$$\forall l \in \overline{1, L}, \quad z_l := \begin{cases} a_l, & \text{If, } a_l \text{ is one of the largest } K \text{ components,} \\ \mathsf{shr}(a_l), & \text{otherwise} \end{cases}$$

Shrinkage Function (Shr)

- $\tau \mathsf{shr}(\tau) \geq 0$, $\forall \tau \in \mathbb{R}$.
- shr acts as a *strict* shrinkage operator over all intervals which do not include 0.
- Any arbitrary function inline with the properties above can be used.
- All the penalized Least-Squares thresholding operators are included.

Generalized Thresholding Rules

Generalized Thresholding (GT) operator: Definition:

For any $\boldsymbol{a} \in \mathbb{R}^L$, $\boldsymbol{z} := T_{\mathsf{GT}}^{(K)}(\boldsymbol{a})$ is obtained coordinate-wise:

$$\forall l \in \overline{1, L}, \quad z_l := \begin{cases} a_l, & \text{If, } a_l \text{ is one of the largest } K \text{ components,} \\ \mathsf{shr}(a_l), & \text{otherwise} \end{cases}$$

In words

- Choose the largest K components of the estimate.
- The rest are shrunk according to the shrinkage rule.

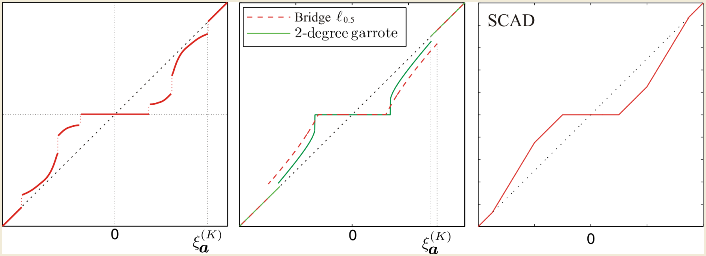
Generalized Thresholding Rules

Generalized Thresholding **(GT)** operator: Definition:

For any $\boldsymbol{a} \in \mathbb{R}^L$, $\boldsymbol{z} := T_{\mathsf{GT}}^{(K)}(\boldsymbol{a})$ is obtained coordinate-wise:

$$\forall l \in \overline{1, L}, \quad z_l := \begin{cases} a_l, & \text{If, } a_l \text{ is one of the largest } K \text{ components,} \\ \mathsf{shr}(a_l), & \text{otherwise} \end{cases}$$

Examples: Generalized Thresholding (GT) operator

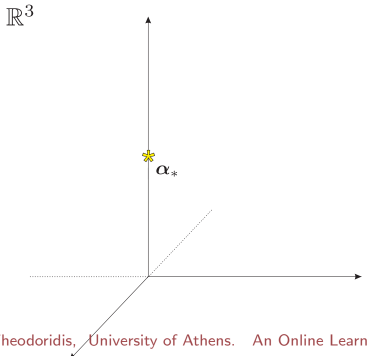


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

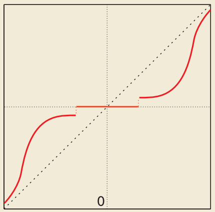
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

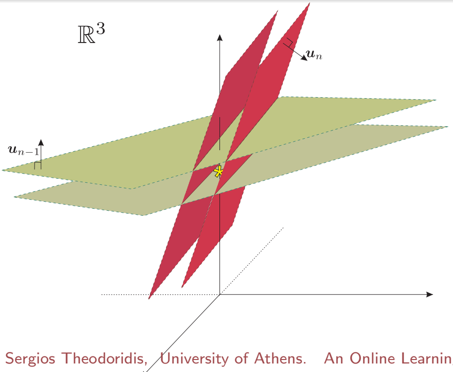


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

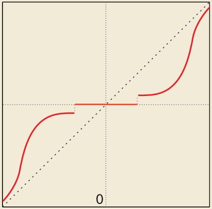
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

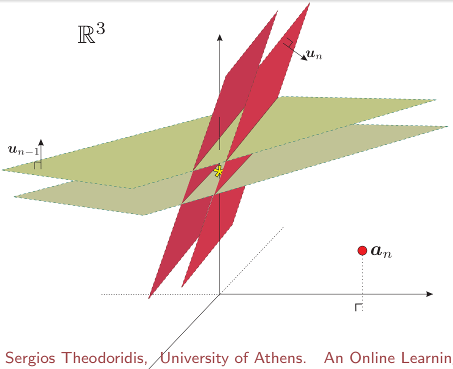


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

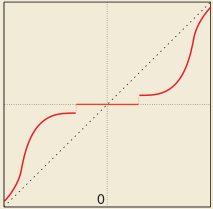
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n\left(\boldsymbol{a}_n + \mu_n\left(\sum_{i=n-q+1}^{n} \omega_i^{(n)}\left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n\right)\right)\right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

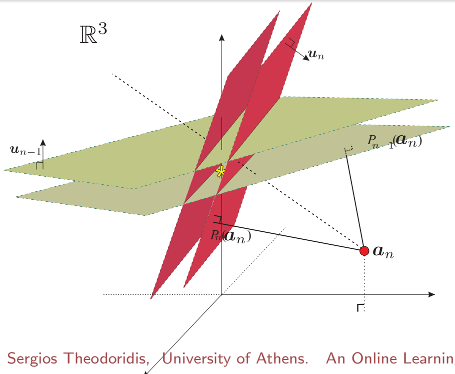


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

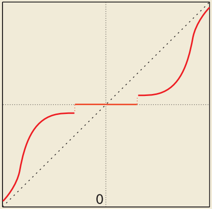
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

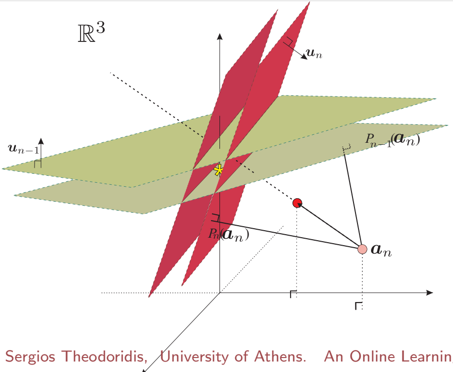


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

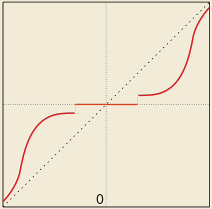
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n\left(\boldsymbol{a}_n + \mu_n\left(\sum_{i=n-q+1}^{n}\omega_i^{(n)}\left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n\right)\right)\right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

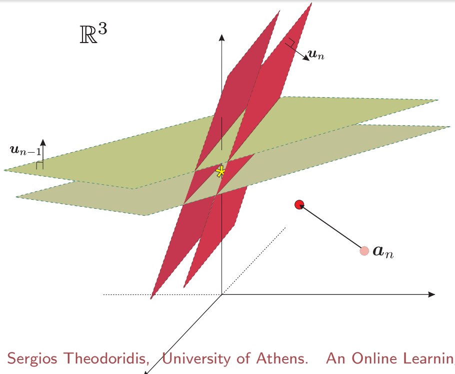


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

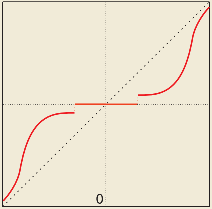
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

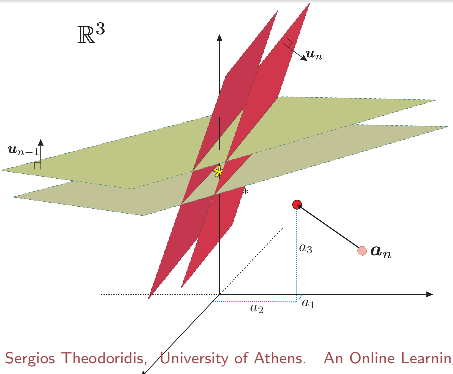


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

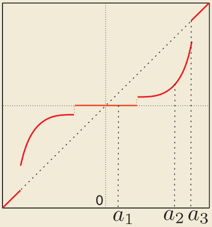
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n\left(\boldsymbol{a}_n + \mu_n\left(\sum_{i=n-q+1}^{n} \omega_i^{(n)}\left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n\right)\right)\right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

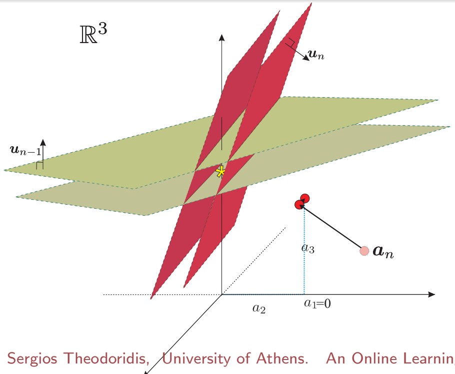


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

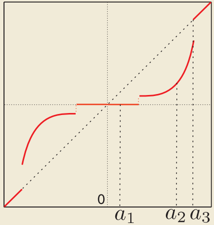
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

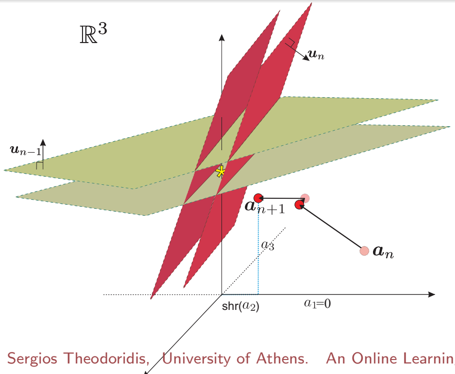


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

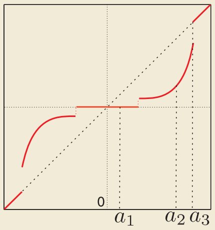
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

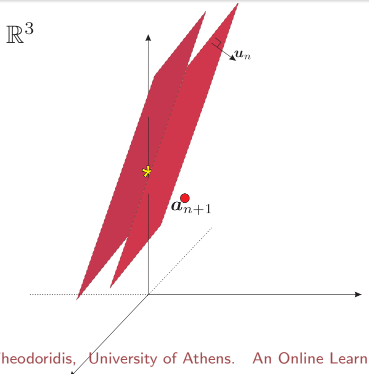


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

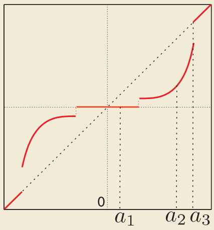
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

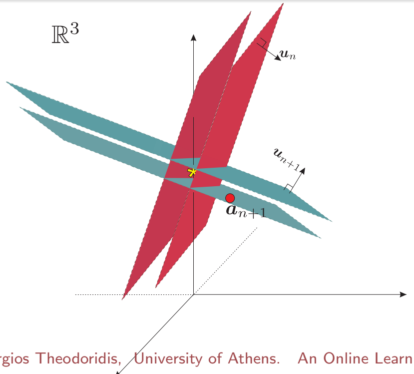


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

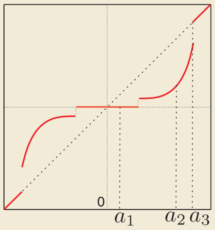
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

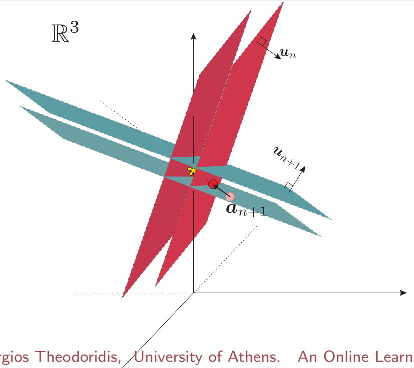


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

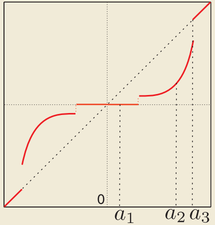
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

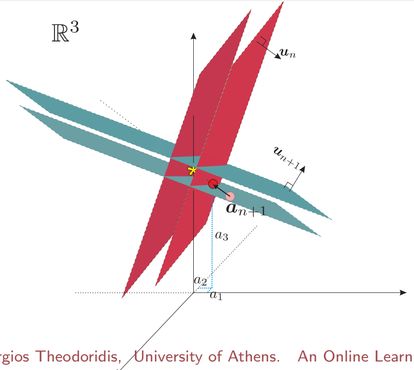


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

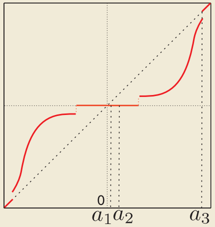
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n \left(\boldsymbol{a}_n + \mu_n \left(\sum_{i=n-q+1}^{n} \omega_i^{(n)} \left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n \right) \right) \right)$$

- Each piece of a-priori information, is also represented by a set



Thresholding Operator

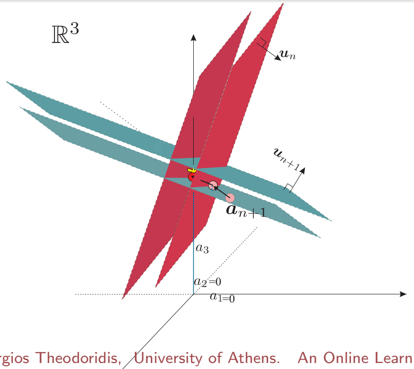


Adaptive Projection-Based Algorithm With Generalized Thresholding (APGT)

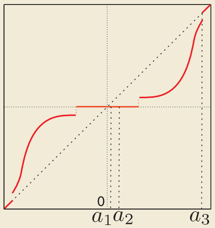
The Algorithm

$$\boldsymbol{a}_{n+1} := T_n\left(\boldsymbol{a}_n + \mu_n\left(\sum_{i=n-q+1}^{n} \omega_i^{(n)}\left(P(\boldsymbol{a}_n) - \boldsymbol{a}_n\right)\right)\right)$$

- Each piece of a-priori information, is also represented by a set



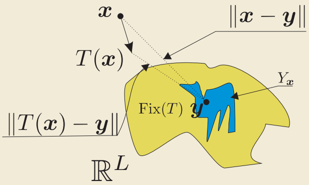
Thresholding Operator



Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

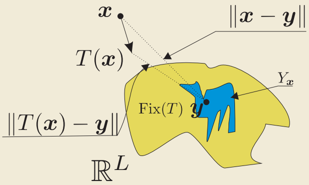
- Partially Quasi-nonexpansive Mapping.
  $\forall \boldsymbol{x} \in \mathbb{R}^L, \exists Y_{\boldsymbol{x}} \subset \mathrm{Fix}(T) : \forall \boldsymbol{y} \in Y_{\boldsymbol{x}},$
  $\|T(\boldsymbol{x}) - \boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$



Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

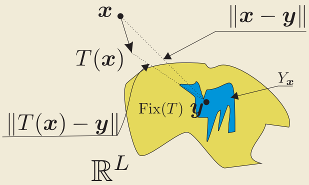
- Partially Quasi-nonexpansive Mapping.
  $\forall \boldsymbol{x} \in \mathbb{R}^L, \exists Y_{\boldsymbol{x}} \subset \mathrm{Fix}(T) : \forall \boldsymbol{y} \in Y_{\boldsymbol{x}},$
  $\|T(\boldsymbol{x}) - \boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$
- The fixed point set of GT is a union of subspaces (non-convex).



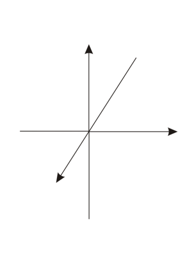
Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

- Partially Quasi-nonexpansive Mapping.
  $\forall \boldsymbol{x} \in \mathbb{R}^L, \exists Y_{\boldsymbol{x}} \subset \mathrm{Fix}(T) : \forall \boldsymbol{y} \in Y_{\boldsymbol{x}},$
  $\|T(\boldsymbol{x}) - \boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$
- The fixed point set of GT is a union of subspaces (non-convex).



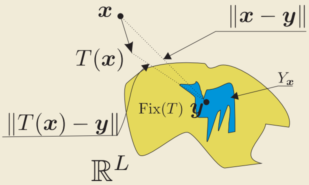
Examples: Union of Subspaces for $s = 2$



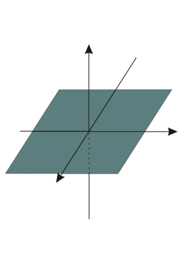
Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

- Partially Quasi-nonexpansive Mapping.
  $\forall \boldsymbol{x} \in \mathbb{R}^L, \exists Y_{\boldsymbol{x}} \subset \mathrm{Fix}(T) : \forall \boldsymbol{y} \in Y_{\boldsymbol{x}},$
  $\|T(\boldsymbol{x}) - \boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$
- The fixed point set of GT is a union of subspaces (non-convex).



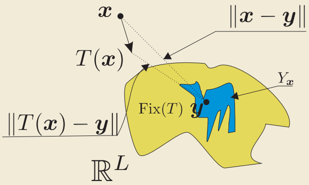
Examples: Union of Subspaces for $s = 2$



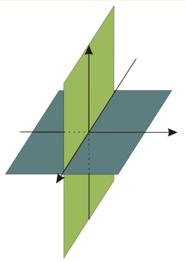
Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

- Partially Quasi-nonexpansive Mapping.
  $\forall \boldsymbol{x} \in \mathbb{R}^L, \exists Y_{\boldsymbol{x}} \subset \mathrm{Fix}(T) : \forall \boldsymbol{y} \in Y_{\boldsymbol{x}},$
  $\|T(\boldsymbol{x}) - \boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$
- The fixed point set of GT is a union of subspaces (non-convex).



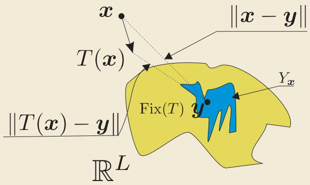
Examples: Union of Subspaces for $s = 2$



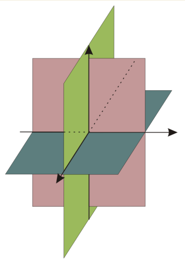
Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

- Partially Quasi-nonexpansive Mapping.
  $\forall \boldsymbol{x} \in \mathbb{R}^L, \exists Y_{\boldsymbol{x}} \subset \mathrm{Fix}(T) : \forall \boldsymbol{y} \in Y_{\boldsymbol{x}}$,
  $\|T(\boldsymbol{x}) - \boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$
- The fixed point set of GT is a union of subspaces (non-convex).



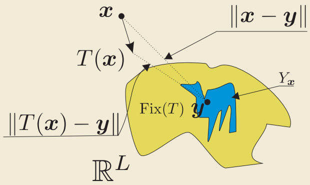
Examples: Union of Subspaces for $s = 2$



Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

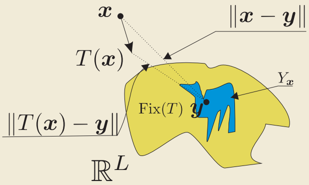
- Partially Quasi-nonexpansive Mapping.
- The fixed point set of GT is a union of subspaces (non-convex).



Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

- Partially Quasi-nonexpansive Mapping.
- The fixed point set of GT is a union of subspaces (non-convex).

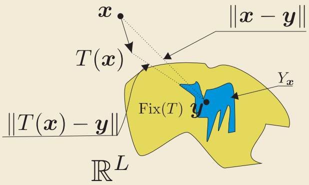


- It has been shown [Slavakis, Kopsinis, Theodoridis, McLaughlin, 2013]:

Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

- Partially Quasi-nonexpansive Mapping.
- The fixed point set of GT is a union of subspaces (non-convex).

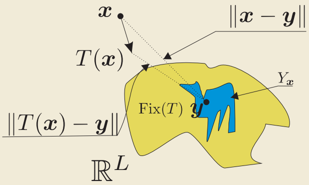


- It has been shown [Slavakis, Kopsinis, Theodoridis, McLaughlin, 2013]:
  - The algorithm leads to a sequence of estimates $(\boldsymbol{a})_{n\in\mathbb{Z}_{\geq 0}}$ whose set of cluster points is nonempty,

Adaptive Projection-Based Algorithm With GT (APGT)

Convergence of APGT

- Partially Quasi-nonexpansive Mapping.
- The fixed point set of GT is a union of subspaces (non-convex).

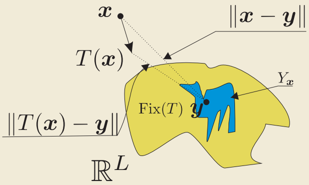


- It has been shown [Slavakis, Kopsinis, Theodoridis, McLaughlin, 2013]:
  - The algorithm leads to a sequence of estimates $(\boldsymbol{a})_{n\in\mathbb{Z}_{\geq 0}}$ whose set of cluster points is nonempty,
  - each one of the cluster points is guaranteed to be, at most, s-sparse,

Adaptive Projection-Based Algorithm With GT (APGT)

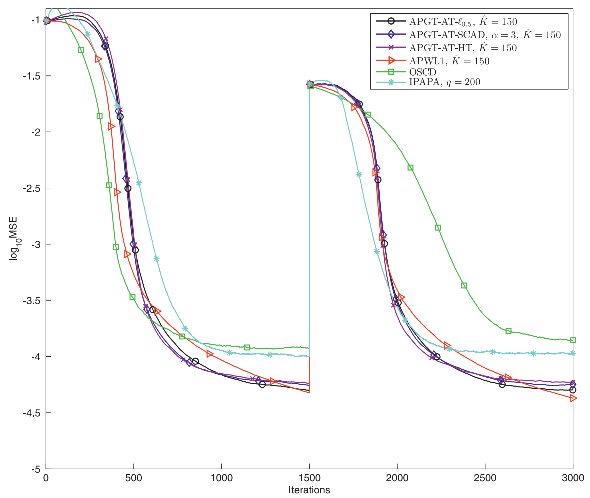
Convergence of APGT

- Partially Quasi-nonexpansive Mapping.
- The fixed point set of GT is a union of subspaces (non-convex).



- It has been shown [Slavakis, Kopsinis, Theodoridis, McLaughlin, 2013]:
  - The algorithm leads to a sequence of estimates $(\boldsymbol{a})_{n\in\mathbb{Z}_{\geq 0}}$ whose set of cluster points is nonempty,
  - each one of the cluster points is guaranteed to be, at most, s-sparse,
  - the solution is located arbitrarily close to an intersection of an infinite number of hyperslabs.

Example: Time-varying case exhibiting an abrupt change

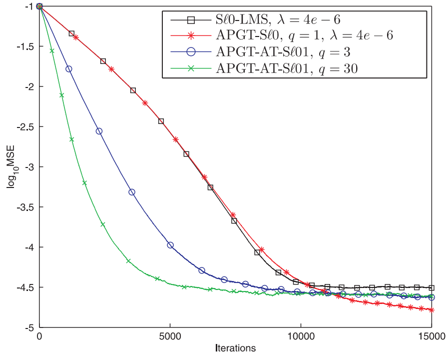


APGT: $\mathcal{O}(qL + qK)$
OSCD: $\mathcal{O}(L^2)$
IPAPA: $\mathcal{O}(q^3)$

$L := 1024$, $s = 100$ (up to $n = 1500$, and $s = 110$ afterwards)

Simulation Examples

Example: Sparse system identification with colored input



$L := 600$, $s = 60$, AR input (cond $\simeq 100$) .

Bibliography

- 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), 2004.
- 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), Nov 2006.
- 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 Trans. Signal Proc.*, vol. 28, Jan. 2011.
- K. Slavakis and I. Yamada. The adaptive projected subgradient method constrained by families of quasi-nonexpansive mappings and its application to online learning. SIAM Journal on Optimization, vol. 23, no. 1, 2013.
- Y. Kopsinis, K. Slavakis, and S. Theodoridis, "Online sparse system identification and signal reconstruction using projections onto weighted ℓ_1 balls," *IEEE Trans. Signal Proc.*, vol. 59, Mar. 2011.
- S. Chouvardas, K. Slavakis, and S. Theodoridis. Adaptive robust distributed learning in diffusion sensor networks. IEEE Transactions on Signal Processing, vol. 59, Oct. 2011.
- S. Chouvardas, K. Slavakis, Y. Kopsinis, and S. Theodoridis. A sparsity promoting adaptive algorithm for distributed learning. IEEE Transactions on Signal Processing, vol. 60, Oct. 2012.
- K. Slavakis, Y. Kopsinis, S. Theodoridis, and S. McLaughlin. Generalized thresholding and online sparsity-aware learning in a union of subspaces. Accepted for publication in the IEEE Transactions on Signal Processing, 2013
- S. Chouvardas, K. Slavakis, S. Theodoridis, and I. Yamada. Stochastic analysis of hyperslab-based adaptive projected subgradient method under bounded noise. IEEE Signal Processing Letters, vol. 20, 2013.
- Y. Kopsinis, K. Slavakis, S. Theodoridis, "Thresholding-Based Online Algorithms of Complexity Comparable to Sparse LMS methods," Under preparation (A part submitted to ISCAS 2013),

- D. Angelosante, J. A. Bazerque, and G. B. Giannakis, 'Online Adaptive Estimation of Sparse Signals: Where RLS Meets the l1-Norm', *IEEE Transactions on Signal Processing*, vol. 58, Jul. 2010.
- E. M. Eksioglu and A. K. Tanc, 'RLS Algorithm With Convex Regularization', *IEEE Signal Processing Letters*, vol. 18, Aug. 2011.
- Y. Chen, Y. Gu, and A. O. Hero, 'Sparse LMS for system identification', *in IEEE International Conference on Acoustics, Speech and Signal Processing,* ICASSP 2009.
- X. Wang and Y. Gu, 'Proof of Convergence and Performance Analysis for Sparse Recovery via Zero-Point Attracting Projection', *IEEE Transactions on Signal Processing*, vol. 60, Aug. 2012.
- G. Mileounis, B.Babadi, N. Kalouptsidis, and V. Tarokh, "An Adaptive Greedy Algorithm with Application to Nonlinear Communications", IEEE Trans. on Signal Processing, vol. 58, Jun. 2010.
- P. Di Lorenzo and A. H. Sayed, "Sparse distributed learning based on diffusion adaptation," IEEE Trans. Signal Processing, vol. 61, March 2013.

"Machine Learning: A Bayesian and Optimization Perspective"

by

Sergios Theodoridis

Academic Press, 2015

1050 pages

Thank you
for your patience...

I hope that there are NO QUESTIONS !!!!!!!!!

Thank you
for your patience...

I hope that there are NO QUESTIONS !!!!!!!!!