AFFINELY CONSTRAINED ONLINE LEARNING AND ITS APPLICATION TO BEAMFORMING

Konstantinos Slavakis

Univ. of Peloponnese, Tripolis 22100, Greece.
Email: slavakis@uop.gr.

Sergios Theodoridis

Univ. of Athens, Athens 15784, Greece.
Email: stheodor@di.uoa.gr.

ABSTRACT

This paper presents a novel method for incorporating a-priori affine constraints in online kernel-based learning tasks. The proposed technique elaborates the generic tool of projections to form a sequence of estimates in Reproducing Kernel Hilbert Spaces (RKHS). The method guarantees that the whole sequence of estimates lies in the given affine constraint set. To validate the algorithm, a beamforming task is considered. The numerical results show that the proposed frame provides with solutions in cases where the classical linear approach collapses, and forms proper beam-patterns as opposed to a recent unconstrained kernel-based regression method.

Index Terms— Learning systems, Beamforming.

1. INTRODUCTION

Kernel methods have become recently the main tool for translating classical linear supervised and unsupervised techniques to nonlinear learning algorithms [1]. The key point of kernel methods is that they offer an efficient way, a nonlinear (implicit) mapping, which "transfers" the processing from a given low dimensional *data space* to the *feature space*; a very high (possibly infinite) dimensional *Reproducing Kernel Hilbert Space (RKHS)* $\mathcal{H}$ [1]. This nonlinear mapping is supplied by a *kernel function* which, basically, defines the feature space $\mathcal{H}$.

Kernel methods have been shown to be highly successful, mainly through batch settings like the celebrated *Support Vector Machine (SVM)* framework [1]. However, batch methods face insurmountable computational obstacles when applied to online adaptive settings, i.e., cases where data arrive sequentially, and the environments exhibit slow variation. Since these obstacles reduce the applicability of kernel methods, recent research focuses on the development of genuine online algorithms [2, 3].

This paper focuses on online kernel methods under the a-priori information that the estimandum satisfies an affine constraint. An algorithmic solution is derived and validated in the context of beamforming. It is demonstrated that with only a few array elements, an enhanced performance is achieved compared to previously used techniques. In other words, we develop an algorithm, of *linear complexity* with respect to the number of unknown parameters, for an affinely constrained optimization task, for online settings, and in infinite dimensional Reproducing Kernel Hilbert Spaces (RKHS).

2. MATHEMATICAL PRELIMINARIES

2.1. Closed convex sets and projection mappings.

We will denote the set of all integers, nonnegative integers, positive integers, real and complex numbers by $\mathbb{Z}$, $\mathbb{Z}_{\geq 0}$, $\mathbb{Z}_{>0}$, $\mathbb{R}$ and $\mathbb{C}$ respectively. Henceforth, the symbol $\mathcal{H}$ will stand for a generally infinite dimensional real Hilbert space equipped with an inner product denoted by $\langle\cdot,\cdot\rangle$. The induced norm becomes $\|\cdot\| := \langle\cdot,\cdot\rangle^{1/2}$.

Given a point $f \in \mathcal{H}$ and a closed convex set $C \subseteq \mathcal{H}$, a way to move from f to a point in C is by means of the *metric projection mapping* P_C *onto* C, which is defined as the mapping that takes f to the *uniquely* existing point $P_C(f)$ of C such that $\|f - P_C(f)\| = \inf\{\|f - f'\| : f' \in C\}$ [4].

2.2. Reproducing Kernel Hilbert Space (RKHS).

Let us consider here a special Hilbert space $\mathcal{H}$. Assume that $\mathcal{H}$ consists of functions defined on $\mathbb{R}^m$, i.e., $f : \mathbb{R}^m \to \mathbb{R}$, for some $m \in \mathbb{Z}_{>0}$. The function $\kappa(\cdot,\cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is called a *reproducing kernel* of $\mathcal{H}$ if $\kappa(\boldsymbol{x},\cdot) \in \mathcal{H}$, $\forall \boldsymbol{x} \in \mathbb{R}^m$, and if $\forall \boldsymbol{x} \in \mathbb{R}^m$ and $\forall f \in \mathcal{H}$, $f(\boldsymbol{x}) = \langle f, \kappa(\boldsymbol{x},\cdot)\rangle$ (*reproducing property*). In this case, $\mathcal{H}$ is called a *Reproducing Kernel Hilbert Space (RKHS)* [1].

Celebrated examples of reproducing kernels are i) the linear kernel (here the associated RKHS is the space $\mathbb{R}^m$ itself [1]), and ii) the Gaussian kernel $\kappa(\boldsymbol{x},\boldsymbol{y}) := \exp(-\frac{(\boldsymbol{x}-\boldsymbol{y})^t(\boldsymbol{x}-\boldsymbol{y})}{2\sigma^2})$, $\forall \boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^m$, where $\sigma > 0$ (here the RKHS is of infinite dimension [1]), and the superscript t stands for transposition.

3. AFFINELY CONSTRAINED ONLINE LEARNING

We assume a sequence of training data $(\boldsymbol{x}_n)_{n\in\mathbb{Z}_{\geq 0}}$ which lies in the Euclidean space $\mathbb{R}^m$. Available is also another sequence of training data $(y_n)_{n\in\mathbb{Z}_{\geq 0}}$ with values in $\mathbb{R}$. Our online learning problem is formed as that of finding an f in an RKHS $\mathcal{H}$ such that the difference $f(\boldsymbol{x}_n) - y_n$ is sufficiently "small" (in some sense), for a large number of indexes n. Moreover, we would like to find a method which also incorporates the additional a-priori information that the desired f belongs also to an affine set $V \subset \mathcal{H}$.

Our exposition evolves along the study of i) the affine constraint set V, and ii) the cost functions formed by the incoming training data.

3.1. The affine constraint set V.

An affine set (or linear variety) V is defined as the translate of a linear subspace M, i.e., there exists a linear subspace M and a $v \in V$ such that $V = v + M$ [4, p. 31]. From now and on, we consider M to be a *closed* linear subspace. The metric projection P_V is given by $P_V(f) = v + P_M(f - v)$, $\forall f \in \mathcal{H}$ [4, p. 25]. The *dimension* of the affine set V is defined as the dimension of the (unique) subspace M [4, p. 248]. The *codimension* of the affine set V is defined as the dimension of the orthogonal complement $M^\perp$ [4, p. 133].

We consider here two examples of affine sets which cover most of the practical situations met in signal processing and machine learning tasks.

Example 1 (Finite dimensional affine set V) In this case, V is the translate of a finite dimensional linear subspace M. In other words, if v_0 is a given point in V and if M is spanned by finite number of elements $\{g_1,\ldots,g_p\}$ in $\mathcal{H}$, then $V := v_0 + M$. The projection P_M has a simple analytic expression given by [4, p. 130]

$$P_M(f) = [g_1,\ldots,g_p]\boldsymbol{G}^\dagger \boldsymbol{c}, \quad \forall f \in \mathcal{H}. \tag{1}$$

In (1) the $p \times p$ matrix $\boldsymbol{G}$, with $\boldsymbol{G}_{ij} := \langle g_i, g_j \rangle$, is a Gram matrix. The symbol † denotes the Moore-Penrose pseudoinverse. The vector $\boldsymbol{c} \in \mathbb{R}^p$ is defined as $\boldsymbol{c} := [\langle f, g_1 \rangle, \ldots, \langle f, g_p \rangle]^t$. Moreover, the notation $[g_1, \ldots, g_p]\boldsymbol{\gamma} := \sum_{i=1}^{p} \gamma_i g_i$, for any p-dimensional vector $\boldsymbol{\gamma}$.

Example 2 (Finite codimensional affine set V) Since the codimension is defined as the dimension of the orthogonal complement $M^{\perp}$, and since our $\mathcal{H}(= M + M^{\perp})$ is, in general, infinite dimensional, this case allows V to have infinite dimensions. Any finite codimensional affine set V can be written as the intersection of a finite number of hyperplanes, i.e., $V = \bigcap_{i=1}^{p} \{f \in \mathcal{H} : \langle f, g_i \rangle = c_i\}$, for some $g_1, \ldots, g_p \in \mathcal{H}$, and some $\boldsymbol{c} := [c_1, \ldots, c_p]^t$ [4, p. 133]. The metric projection P_V is given by the following simple analytic formula: $\forall f \in \mathcal{H}$,

$$P_V(f) = f + [g_1, \ldots, g_p]\boldsymbol{G}^{\dagger}(\boldsymbol{c}^t - [\langle f, g_1 \rangle, \ldots, \langle f, g_p \rangle]^t). \quad (2)$$

In order to get P_M, set $\boldsymbol{c}$ equal to $\boldsymbol{0}$ in (2) (For a proof of (2) see [3, Appendix A]).

3.2. Cost functions: the effect of the training data.

We will use now the information carried by the training sequences $(\boldsymbol{x}_n)$ and (y_n) to form a sequence of cost objectives. It turns out that this is equivalent to an infinite sequence of certain closed convex constraints.

Our goal is to estimate $f(\cdot)$ so that the distance of $f(\boldsymbol{x}_n)$ to y_n is "small" enough, for as many n as possible. To quantify this rationale, we adopt the following sequence of cost functions, widely used in robust statistics [1]: given a small positive $\epsilon > 0$, let $\forall n \in \mathbb{Z}_{\geq 0}$,

$$\Theta_n(f) := \max\{0, |f(\boldsymbol{x}_n) - y_n| - \epsilon\}, \quad \forall f \in \mathcal{H}. \quad (3)$$

For each $n \in \mathbb{Z}_{\geq 0}$, the set of minimizers of the *non-differentiable* cost function (3) is obviously the set

$$S(n) = \{f \in \mathcal{H} : |\langle f, \kappa(\boldsymbol{x}_n, \cdot) \rangle - y_n| \leq \epsilon\}, \quad (4)$$

where the inner product in (4) comes from the fundamental reproducing property of the RKHS $\mathcal{H}$ seen in Section 2.2 and (3). The set $S(n)$ in (4) is a closed convex set, widely known as *hyperslab*. For an illustration of a hyperslab see Fig. 1.

The projection mapping $P_{S(n)}(f)$ has a simple and analytic form. It is also of *linear complexity with respect to the number of the kernel functions* used for the representation of f. To express the formula in a compact form, we introduce the following coefficients (the proof is omitted due to lack of space): $\forall n \in \mathbb{Z}_{\geq 0}$,

$$\beta_n := \begin{cases} \frac{y_n - \epsilon - \langle f, \kappa(\boldsymbol{x}_n, \cdot) \rangle}{\kappa(\boldsymbol{x}_n, \boldsymbol{x}_n)}, & \text{if } \langle f, \kappa(\boldsymbol{x}_n, \cdot) \rangle < y_n - \epsilon, \\ 0, & \text{if } |\langle f, \kappa(\boldsymbol{x}_n, \cdot) \rangle - y_n| \leq \epsilon, \\ \frac{y_n + \epsilon - \langle f, \kappa(\boldsymbol{x}_n, \cdot) \rangle}{\kappa(\boldsymbol{x}_n, \boldsymbol{x}_n)}, & \text{if } \langle f, \kappa(\boldsymbol{x}_n, \cdot) \rangle > y_n + \epsilon. \end{cases} \quad (5)$$

Then, $P_{S(n)}(f) = f + \beta_n \kappa(\boldsymbol{x}_n, \cdot)$, $\forall n \in \mathbb{Z}_{\geq 0}$, $\forall f \in \mathcal{H}$.

It is well-known, as in the celebrated Affine Projection Algorithm (APA) [3], that concurrent processing can increase the speed of convergence of an algorithm. As such, at each iteration index n, we consider a convex combination of loss functions (3) corresponding to the hyperslabs of previous time instants, $n, n-1, \ldots, n-q+1$. This has a smoothing effect on the result. To this end, let the index set

$$\mathcal{J}_n := \begin{cases} \overline{0, n}, & \text{if } n < q - 1, \\ \overline{n - q + 1, n}, & \text{if } n \geq q - 1, \end{cases} \quad (6)$$

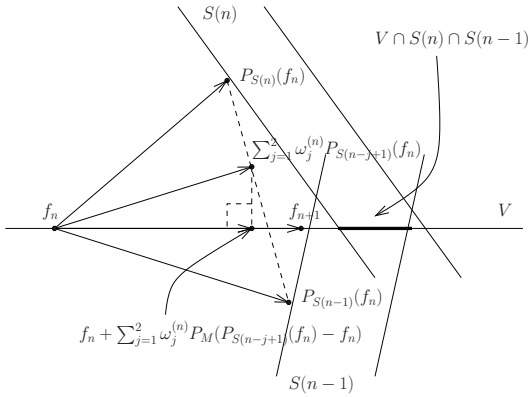

Fig. 1. Illustration of the proposed algorithm (7). For simplicity, we consider here only two hyperslabs: $S(n)$ and $S(n-1)$. The set of all convex combinations $\sum_{j \in \mathcal{J}_n} \omega_j^{(n)} P_{S(j)}(f_n)$ in (7) is denoted by the dotted line. However, the extrapolation given by the parameter μ_n takes us closer to the thick black line segment which denotes the desired functions at index n. As time goes by, and due to the online nature of the problem, the hyperslabs change according to the incoming sequence of data, while the affine set V remains fixed. The generated sequence of estimates (f_n) given in (7) lies in V.

where the symbol $\overline{j_1, j_2} := \{j_1, j_1 + 1, \ldots, j_2\}$, for any integers $j_1 \leq j_2$, where q is a predefined positive integer. To quantify the contribution of each hyperslab to such concurrent processing, we assign a convex weight $\omega_j^{(n)}$, i.e., $\omega_j^{(n)} \in [0, 1)$, and $\sum_{j \in \mathcal{J}_n} \omega_j^{(n)} = 1$, to each $j \in \mathcal{J}_n$.

To summarize, *we seek for a point f in the intersection of the following infinite collection of closed convex sets: $V \cap (\bigcap_{n=n_0}^{\infty} S(n))$, for, let's, say some nonnegative integer n_0.*

4. THE ALGORITHM

For any $f_0 \in V$, form the following sequence $\forall n \in \mathbb{Z}_{\geq 0}$,

$$f_{n+1} := f_n + \mu_n \sum_{j \in \mathcal{J}_n} \omega_j^{(n)} P_M(P_{S(j)}(f_n) - f_n), \quad (7)$$

where $\mathcal{J}_n$ is the index set of (6). In the case of Example 1, let $f_0 := v_0$. In the case of Example 2 let for example $f_0 := P_V(0)$. The algorithmic solution (7) is based on the *Adaptive Projected Subgradient Method (APSM)* [5]. The parameter μ_n is explained below. A geometric illustration of (7) is depicted in Fig. 1. Since the projections involved in (7) are of linear computational load, the proposed algorithm is of *linear complexity* with respect to the number of the used kernel functions.

Due to the concurrency, i.e., the multiplicity of the constraints that are processed at every n via the index set $\mathcal{J}_n$, the relation (7) extrapolates the projections mappings $\{P_{S(j)}\}_{j \in \mathcal{J}_n}$. The range of the *extrapolation parameter* μ_n, which affects the speed of convergence, is calculated recursively within an iterative scheme as follows: given the current estimate f_n, μ_n takes values inside the interval $[0, 2\mathcal{M}_n]$, where $\mathcal{M}_n$ is calculated by the following closed form:

$$\mathcal{M}_n := \begin{cases} \frac{\sum_{j \in \mathcal{J}_n} \omega_j^{(n)} \|P_{S(j)}(f_n) - f_n\|^2}{\|\sum_{j \in \mathcal{J}_n} \omega_j^{(n)} P_M(P_{S(j)}(f_n) - f_n)\|^2}, \\ \quad \text{if } \sum_{j \in \mathcal{J}_n} \omega_j^{(n)} P_M(P_{S(j)}(f_n) - f_n) \notin M^{\perp}, \\ 1, \text{ otherwise}, \end{cases} \quad (8)$$

where its complexity is of order $\mathcal{O}(q^2)$ (the computational load is shared with (7)). By the convexity of the function $\|\cdot\|^2$ and the def-

inition of $\mathcal{M}_n$, we can derive that $\mathcal{M}_n \geq 1$. As such, μ_n can take values ≥ 2. In general, the larger the μ_n, but with $\mu_n \leq 2\mathcal{M}_n$, the larger the extrapolation step in (7), which potentially accelerates the algorithm. For the special case where $\mathcal{J}_n$ contains only a single element, i.e., $\mathcal{J}_n := \{n\}$, then we no longer have concurrency.

If we recall the definition of an affine set V as the translate of a linear subspace M, i.e., $V = v + M$, for any $v \in V$, then it is easy to see by induction, and the fact that $f_0 \in V$, that for each iteration, *the sequence of estimates in* (7) *lie in* V*, i.e.,* $(f_n) \subset V$.

Under mild conditions, the algorithm introduced in (7) possesses a number of desirable theoretical properties: monotone approximation, strong convergence to a point that belongs both to V and to an infinite sequence of hyperslabs (4), asymptotic minimization of the sequence of cost functions (3), etc [5].

5. SPARSIFICATION

Let us give, now, an alternative expression for the sequence of estimates formed by (7). By using mathematical induction, one can show (the proof is omitted due to lack of space) that

$$f_{n+1} = \sum_{i=1}^{p} \gamma_i^{(n+1)} g_i + \sum_{j=0}^{n} \eta_j^{(n+1)} \kappa(\boldsymbol{x}_j, \cdot), \forall n, \qquad (9)$$

where the $\{g_i\}_{i=1}^p$ describe V (see Section 3.1), and where $\{\gamma_i^{(n+1)}\}_{i=1}^p$, $\{\eta_j^{(n+1)}\}_{j=1}^n$ are appropriately defined real coefficients. This is basically a generalization of the celebrated Representer Theorem, where, here, the first of the two sums is due to the affine constraint.

It is obvious that as the index n advances, more and more kernel functions enter the series in (9), making the memory requirements and the computational load to grow unbounded. This phenomenon is shared by all online (sequential) kernel methods [2]. Hence, one of the main issues in online methods is the sparsification of the kernel series, such as the one appearing in (9).

Here we focus on the sparsification procedure introduced in [2] and generalized in [3]. The method relies on the construction of a basis $\mathcal{B}_n$, i.e., a set of linearly independent kernel functions, for every index n. The way to update the basis is based on fundamental linear algebra properties: whenever the incoming kernel function $\kappa(\boldsymbol{x}_n, \cdot)$ is *approximately linearly independent* to the already existing basis $\mathcal{B}_{n-1}$, the basis is augmented by inserting $\kappa(\boldsymbol{x}_n, \cdot)$ into $\mathcal{B}_{n-1}$: $\mathcal{B}_n := \mathcal{B}_{n-1} \cup \{\kappa(\boldsymbol{x}_n, \cdot)\}$. Otherwise, i.e., whenever the incoming kernel $\kappa(\boldsymbol{x}_n, \cdot)$ is *approximately linearly dependent* to the existing $\mathcal{B}_{n-1}$, the basis remains the same: $\mathcal{B}_n := \mathcal{B}_{n-1}$. The representation of the kernel $\kappa(\boldsymbol{x}_n, \cdot)$ as a linear combination of the basis $\mathcal{B}_n$ elements will be denoted by $\pi_n(\kappa(\boldsymbol{x}_n, \cdot))$ [3]. For details of the sparsification method we refer the reader to [2, 3]. This sparsification method is of quadratic complexity with respect to the cardinality of the basis $\mathcal{B}_n$. However, we stress here that besides this technique, other approximating methods of *linear complexity* can be employed for the sparsification of (9), e.g., see [6].

We apply now the sparsification technique to obtain an approximate version of the algorithm (7) as follows. Recall from Section 3.2 that $P_{S(j)}(f_n) - f_n = \beta_j^{(n)} \kappa(\boldsymbol{x}_j, \cdot)$. Then, for any $f_0 \in V$,

$$\tilde{f}_{n+1} := \tilde{f}_n + \tilde{\mu}_n \sum_{j \in \mathcal{J}_n} \omega_j^{(k)} \tilde{\beta}_j^{(n)} P_M(\pi_n(\kappa(\boldsymbol{x}_j, \cdot))), \forall n, \qquad (10)$$

where $\tilde{\mu}_n$ and $\tilde{\mathcal{M}}_k$ are defined as in Section 4.

6. APPLICATION TO BEAMFORMING

We consider the system of Fig. 2. The *steering vector* associated with a planar wave of wavelength λ, arriving to a Uniform Linear Array (ULA) with an angle $\theta \in [0, \pi]$, called the *Direction Of Arrival (DOA)*, is defined as [7] $\boldsymbol{s} := \boldsymbol{s}(\theta, d, \lambda) := [1, e^{2\pi i \frac{d}{\lambda} \cos\theta}, \ldots, e^{2\pi i (N-1) \frac{d}{\lambda} \cos\theta}]^t \in \mathbb{C}^N$, where $i := \sqrt{-1}$. The steering vector corresponding to the *Signal Of Interest (SOI)* will be denoted by $\boldsymbol{s}_0$, while $\boldsymbol{s}_j$ associates to the j-th jammer, $\forall j \in \overline{1, J}$, for $J \in \mathbb{Z}_{>0}$ (given integers $j_1 \leq j_2$, define $\overline{j_1, j_2} := \{j_1, j_1 + 1, \ldots, j_2\}$).

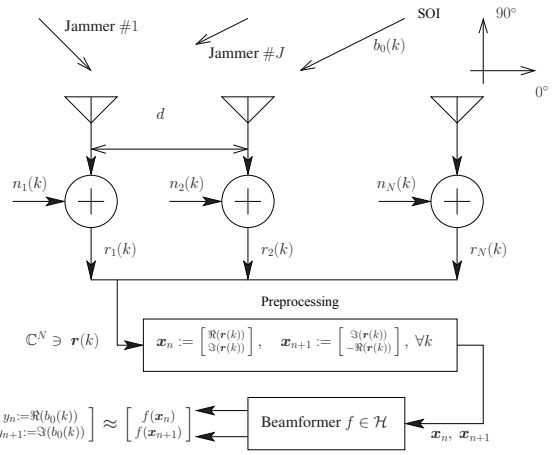

Fig. 2. A Uniform Linear Antenna (ULA) of N elements, with an interelement distance $d > 0$. The beamformer is the function f which belongs to a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. The index $k \in \mathbb{Z}_{\geq 0}$ counts the number of snapshots received by the ULA. Its connection with the index n is given by the relation $n := 2k$, $\forall k \in \mathbb{Z}_{\geq 0}$. To work with real vectors and to accommodate complex input signals, a preprocessing step is applied prior the signal enters the beamformer. The output is a two-dimensional real vector comprising from the real part $\Re(\cdot)$ and the imaginary part $\Im(\cdot)$ of the beamformer's output.

The signal $(\boldsymbol{r}(k))_{k \in \mathbb{Z}_{\geq 0}}$ received by the ULA is given by $\boldsymbol{r}(k) := \sum_{l=0}^{J} \alpha_l b_l(k) \boldsymbol{s}_l + \boldsymbol{n}(k)$, $\forall k \in \mathbb{Z}_{\geq 0}$. The complex scalar random processes $(b_l(k))_{k \in \mathbb{Z}_{\geq 0}}$, $l \in \overline{0, J}$, denote the symbols carried by the SOI and the jammers. The BPSK modulation scheme is used here, so that $b_l(k) = \pm 1$. The complex vector random process $(\boldsymbol{n}(k))_{k \in \mathbb{Z}_{\geq 0}} \subset \mathbb{C}^N$ stands for the additive noise. The complex coefficients $\alpha_l \in \mathbb{C}$, $l \in \overline{0, J}$, comprise a variety of parameters like the signal power, channel attenuation, etc. In order to work with real vectors, a complex vector in $\mathbb{C}^N$ is mapped to a vector in $\mathbb{R}^{2N}$, by using its real $\Re(\cdot)$ and its imaginary $\Im(\cdot)$ part, as shown in Fig. 2.

Here, we consider a limited number of array elements $N := 3$ with $d/\lambda := 0.5$. The SOI's DOA is 90°, and the DOAs of five jammers are 30°, 80°, 95°, 130°, 160°. Gaussian i.i.d. noise is used to form the process $(\boldsymbol{n}(k))$. The SNRs are given by 10, 20, 30, 30, 10, and 30 dB for the SOI and the jammers respectively. We used the Gaussian kernel function for working in an infinite dimensional RKHS $\mathcal{H}$ (see Section 2.2), with $\sigma^2 := 0.5$ (The method appears to be insensitive to small variations of the value of σ^2).

A large variety of constraints for the classical linear beamforming scenario are formed as affine sets [7]. The same rationale can be used also in our nonlinear strategy. For example, given a steering vector $\boldsymbol{s} \in \mathbb{C}^N$, one wishes the output of the beamformer, when receiving from a transmitter located at $\boldsymbol{s}$, to be equal to a predefined complex number e. If we define $\hat{\boldsymbol{s}}_1 := [\Re(\boldsymbol{s}), \Im(\boldsymbol{s})]^t$ and $\hat{\boldsymbol{s}}_2 := [\Im(\boldsymbol{s}), -\Re(\boldsymbol{s})]^t$, as in Fig. 2, as well as $c_1 := \Re(e)$ and $c_2 := \Im(e)$, then our wish takes the form $f(\hat{\boldsymbol{s}}_1) = c_1$ and $f(\hat{\boldsymbol{s}}_2) = c_2$. If we

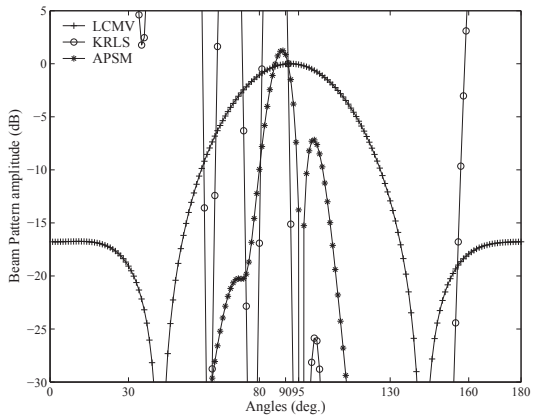

Fig. 3. Since both APSM and KRLS are nonlinear techniques, the concept of "beam-pattern" is slightly different than the classical one for linear methods. Notice that due to nonlinearity, the superposition of beam-patterns is no longer feasible. Hence, here, by "beam-pattern" we mean the ability of a technique to form a solution that satisfies the a-priori constraints imposed by the designer. The KRLS offers unconstrained nonlinear regression, so it naturally shows inability to control the beam-pattern of the ULA. The classical linear approach of LCMV needs more antenna elements, i.e., more degrees of freedom, to produce a narrower main lobe. The proposed affinely constrained optimization approach of the APSM satisfies the affine set constraint as can be seen by the sharp valley at the DOA of 95°, and the SINR value in Table 1.

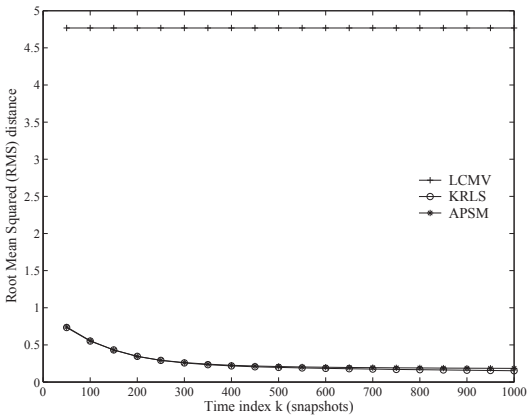

Fig. 4. The average Root Mean Squared (RMS) distance to the BPSK symbols ± 1 in the plane $\mathbb{R}^2$. These curves depict also the convergence properties of the algorithms. Two variants of the sparsification method of Section 5 were used by controlling the parameter ν [2]. The value $\nu = 0.25$ resulted into a basis with average cardinality 1789.2, while the value $\nu = 0.5$ gave 1152.56 basis vectors. However, all of the beam-patterns and the RMS distances were almost similar for these two variants.

	Input	LCMV	KRLS	APSM
SINR (dB)	-24.93	-22.57	Very low	21.01

Table 1. This table illustrates the various Signal to Interference and Noise Ratios (SINR) for the elaborated methods. As in Fig. 3, the SINR concept is slightly different here, for the nonlinear methods APSM and KRLS, than the standard one for the linear techniques. By SINR, and for the nonlinear APSM and KRLS, we mean the difference in dB of the outputs of the array when "pointed to" directions occupied by the respective transmitters. In other words, it measures the beam-pattern formation in Fig. 3. The KRLS, as an unconstrained kernel-based regression approach puts no effort on the beam-pattern design, and resulted into a very low negative value for the SINR.

recall the fundamental reproducing property of the RKHS $\mathcal{H}$ in Section 2.2, then we form the following affine constraint set: $V := \bigcap_{l=1}^{2}\{f \in \mathcal{H} : \langle f, \kappa(\hat{\boldsymbol{s}}_l, \cdot)\rangle = c_l\}$. In other words, $g_1 := \kappa(\hat{\boldsymbol{s}}_1, \cdot)$ and $g_2 := \kappa(\hat{\boldsymbol{s}}_2, \cdot)$ in Example 2. Let's say that we wish the array to have output 1 at the DOA of 90°, and 0 at 95°.

We choose $q := 20$ for the index set $\mathcal{J}_n$ (6), and all the convex weights $\{\omega_j^{(n)}\}$ are set equal to each other for every index n. The extrapolation parameter $\mu_n := 1.95\mathcal{M}_n$, $\forall n \in \mathbb{Z}_{\geq 0}$. We let also $\epsilon := 0.1$ in (4) (The method appears to be insensitive to small variations of the value of ϵ. Moreover, for large changes, the behaviour of the design was seen to be similar). The total number of complex training data was set equal to 1000, a separate set of 200 complex test data were used to validate the obtained results, and a set of 500 complex data were used to calculate the corresponding SINRs. For each realization of the experiment, we calculate the Root Mean Squared (RMS) distance of the beamformer's output to the BPSK symbols ± 1, and the array beam-patterns. We performed 100 realizations and uniformly averaged the results.

The proposed method (denoted by APSM) is compared to the classical *Linearly Constrained Minimum Variance (LCMV)* beamformer [7], and the nonlinear unconstrained regression approach realized by the *Kernel Recursive Least Squares (KRLS)* algorithm [2]. The results are depicted in Figs. 3, 4, and Table 1.

7. CONCLUSIONS

This paper presented a novel online kernel-based learning method for tasks where there is the a-priori knowledge that the desired function satisfies an affine constraint in the RKHS. The proposed method generates a sequence of estimates which lies in the affine constraint. The method is applied to the beamforming task, and it is shown that only with a few array antenna elements, it outperforms significantly the linear beamforming approach, and forms proper beam-patterns as opposed to a recently unconstrained nonlinear regression technique.

8. REFERENCES

[1] B. Schölkopf and A. J. Smola, *Learning with Kernels*, MIT Press, Cambridge, MA, 2001.

[2] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Proc.*, vol. 52, no. 8, pp. 2275–2285, Aug. 2004.

[3] K. Slavakis and S. Theodoridis, "Sliding window generalized kernel affine projection algorithm using projection mappings," *EURASIP Journal on Advances in Signal Processing*, 2008, 16 pages, doi:10.1155/2008/735351.

[4] F. Deutsch, *Best approximation in inner product spaces*, Springer-Verlag, New York, 2001.

[5] I. Yamada and N. Ogura, "Adaptive Projected Subgradient Method for asymptotic minimization of sequence of nonnegative convex functions," *Numerical Functional Analysis and Optimization*, vol. 25, no. 7&8, pp. 593–617, 2004.

[6] K. Slavakis, S. Theodoridis, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Trans. Signal Proc.*, vol. 56, no. 7, pp. 2781–2796, 2008.

[7] H. L. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory*, John Wiley & Sons, New York, 2002.