

Online Tracking of Linear Subspaces

Koby Crammer

Department of Computer and Information Science, University of Pennsylvania
crammer@cis.upenn.edu

Abstract. We address the problem of online de-noising a stream of input points. We assume that the clean data is embedded in a linear subspace. We present two online algorithms for tracking subspaces and, as a consequence, de-noising. We also describe two regularization schemas which improve the resistance to noise. We analyze the algorithms in the loss bound model, and specify some of their properties. Preliminary simulations illustrate the usefulness of our algorithms.

1 Introduction and Problem Setting

Subspace analysis and subspace tracking (e.g. [1]) are important tools in various adaptive signal processing tasks, such as bearing estimation [2] and beamforming [3]. Mathematically, the algorithm receives a sequence of input vectors and returns a linear subspace that describes the data well. Assuming that the data consist of points from a low-dimensional subspace corrupted with isotropic noise which pulled it out of the original subspace [4, 5], the reconstructed subspace can be used to clear or filter the noisy data.

We present online algorithms for subspace tracking and analyze them in the loss bound model. Unlike previous analysis for these types of algorithms (e.g. [1]), we do not use any statistical assumptions over the source of the input points. The goal of the learning algorithm is to de-noise new data points by identifying this subspace. Given a data point, the algorithm is required to output the underlying uncorrupted point. Specifically, we measure the performance of the algorithm relative to the *uncorrupted* version of each point, rather than the corrupted observed version. The algorithms we present can also track drifting or switching subspaces.

The tracking subspace problem shares common properties with both multivariate regression and one-class classification [6]. As in regression, the learner maintains a function (linear transformation) and outputs a vector for a given input vector. In this view, our problem is a regression from a vector space to itself that uses the class of positive semidefinite (PSD) matrices as regressors. This approach to subspace analysis [7] was used to devise sparse principal component analysis (PCA).

Similarly to one-class learning, the learner is *not* exposed to a feedback signal or to a “right” answer. The only information at hand are the input points. The goal of a one-class learner is to identify a meaningful *subset* in space, in the sense that it captures most, if not all of the points. Similarly, our primary goal is to find a meaningful *vector subspace* which contains most of the *weight* of the points. An additional similarity between one-class learning and subspace tracking is that in both cases there is a trivial solution. Given any set of points, one can always find a convex body that encloses all of

them, and we can always find a vector subspace that contains all of their weight (using the identity matrix). In this aspect both problems are ill-defined.

We now describe the subspace tracking problem formally. Let $\mathbf{x}_1 \dots \mathbf{x}_m \in \mathbb{R}^d$ be a set of column vectors. We view the problem as a filtering problem [4, 5], $\mathbf{x}_i = \mathbf{y}_i + \boldsymbol{\nu}_i$, where \mathbf{y}_i lies in a low dimensional linear subspace and $\boldsymbol{\nu}_i$ is the unknown noise for this point. Since the goal is to track the linear subspace we assume that there exists an unknown target idempotent projection matrix Q such that $\mathbf{y}_i = Q\mathbf{x}_i$ and $\boldsymbol{\nu}_i = (I - Q)\mathbf{x}_i$. That is, the noise is taken to be orthogonal to the clean data, because we cannot separate noise components projected by Q onto the subspace and input points.

PCA computes an orthogonal set $A \in \mathbb{R}^{d \times n}$ of n vectors, which is the n eigenvectors corresponding to the top n eigenvalues of the covariance matrix, $\sum_i \mathbf{x}_i \mathbf{x}_i^T$. This basis is often used for compression since each point $\mathbf{x} \in \mathbb{R}^d$ is represented using n values $A^T \mathbf{x}$. In addition, PCA can be used for de-noising with the matrix $Q = AA^T$. Since A is composed of orthonormal vectors, Q is a *projection*, that is, it is symmetric, positive semidefinite (PSD) and idempotent (its eigenvalues are either zero or one). In this paper we adopt this view of PCA, and focus on learning matrices P of this form. Unlike PCA, we do not reduce explicitly the number of components of a vector (by using the matrix A). In other words, we seek a low-dimensional subspace, but represent data in the original vector space of dimension d . Since the restriction that the eigenvalues will be *either* zero or one is algorithmically challenging because it involves integer programming, we relax the idempotency assumption. Our learning algorithms seek linear transformations which are symmetric and positive semidefinite. We refer to these transformations (P or P_i) as projections. When the projections are also idempotent (i.e. all eigenvalues are either zero or one), we will refer to them as *idempotent projections* (Q). One of the algorithms described below always maintains a linear transformation with eigenvalues *between* zero and one. This is often considered the natural relaxation of idempotency.

We present and analyze two *online* learning algorithms for filtering through subspace tracking. Both algorithms can also be used to track *non-stationary* sequences. The first algorithm is motivated by a *gradient descent* approach and the second by an *Euclidean projection*¹. We use the loss-bound model of online learning to analyze the algorithms. The algorithms we consider work in rounds. On round i an online learning algorithm chooses a linear subspace represented by a PSD matrix P_i . It then receives a point \mathbf{x}_i , outputs the projection of \mathbf{x}_i onto the chosen subspace and suffers loss which is a function of the discrepancy between the projection $P_i \mathbf{x}_i$ and the clean point $Q\mathbf{x}_i$, i.e. $\ell(P_i \mathbf{x}_i, Q\mathbf{x}_i)$. Finally, the subspace representation is updated and the next round follows. Note that Q or $Q\mathbf{x}_i$ are *unrevealed* to the learner algorithm, which makes the learning task more involved. We use the matrix Q only for analysis.

Previous work on learning PSD matrices falls into two kinds. The first kind of algorithm builds a general symmetric matrix which is either restricted to be PSD (e.g. [8]) or in a second step projected back on the PSD cone [9]. The second kind of algorithm [10], employ the costly operation of matrix exponentiation which automatically yields PSD matrices. The former approaches employ loss functions which are often linear in

¹ We use the term projection in two ways. First, throughout the paper it refers to a symmetric PSD linear transformation. Second, we use the projection operation to derive the second of the two online algorithms.

the matrix, while the latter uses a loss which is quadratic in the matrix. In this work, we have the benefit of both approaches. Our algorithms are both very simple, involve only addition operations and maintain matrices which are guaranteed to be PSD with no additional operations, even though the quadratic loss is used.

Notation: For a matrix P , the property of being a positive semi-definite matrix is denoted by $P \succeq 0$. Given a vector \mathbf{x} , we denote by $X = \mathbf{x}\mathbf{x}^\top$ the outer-product of \mathbf{x} with itself. A unit vector is denoted by $\hat{\mathbf{x}} = \mathbf{x}/\|\mathbf{x}\|$, and $\hat{X} = \hat{\mathbf{x}}\hat{\mathbf{x}}^\top$ is a rank-one symmetric matrix with eigenvalue 1. Finally, $\|P\|_p$ is ℓ_p norm of the vector generated by concatenating the columns of matrix P .

2 Gradient Algorithm

We start with the description of an online algorithm based on gradient descent. After an input point \mathbf{x}_i has been observed we wish to update our current subspace (represented by P_i) based on this point. Since there is no corresponding feedback signal, we have no choice but to use the point itself as a guide, so we seek to decrease the loss $\ell(\mathbf{x}_i, P\mathbf{x}_i)$. This only approximates our true loss, but as we shall see in the sequel, it is enough. However, we do not want to make big changes from our current subspace, as it captures our knowledge of previous examples. Therefore, we define the following update,

$$P_{i+1} = \arg \min_P \frac{1}{2} \|P - P_i\|^2 + \alpha \ell(\mathbf{x}_i, P\mathbf{x}_i) \quad \text{s.t.} \quad P = P^T, P \succeq 0. \quad (1)$$

where $\alpha > 0$ is a trade-off parameter. In this section we focus in the squared loss,

$$\ell(\mathbf{x}_i, P\mathbf{x}_i) = \frac{1}{2} \|\mathbf{x}_i - P\mathbf{x}_i\|^2. \quad (2)$$

The two constraints ensure that the eigenvalues of P_{i+1} are positive real numbers. Thus, similarly to PCA we will be able to reduce the dimension by performing eigen-decomposition. We derive the update rule for the algorithm by solving the optimization problem. For now we omit the PSD constraint in Eq. (1). We show below that the solution of the optimization problem is in fact PSD with bounded eigenvalues.

The Lagrangian of the optimization problem defined by Eq. (1) is,

$$\mathcal{L}(P; Z) = \frac{1}{2} \|P - P_i\|^2 + \alpha \frac{1}{2} \|\mathbf{x}_i - P\mathbf{x}_i\|^2 - \text{Tr} [Z(P - P^T)]. \quad (3)$$

To solve the problem we first differentiate \mathcal{L} with respect to P and set the result to zero,

$$P_{i+1} - P_i - \alpha X_i + \alpha P_{i+1} X_i - Z^T + Z = 0. \quad (4)$$

As we shall see in Sec. 3 we can solve Eq. (4) analytically, but it involves non-linear terms arising from matrix inversion. Instead, we use the fact that, for reasonable small values of α , the matrices $P_{i+1} X_i$ and $P_i X_i$ are close to each other. We thus approximate Eq. (4) by,

$$P_{i+1} = P_i + \alpha (X_i - P_i X_i) + \tilde{Z}, \quad (5)$$

where we define the anti-symmetric matrix $\tilde{Z} = Z^T - Z$. We eliminate \tilde{Z} from the solution by enforcing the symmetry constraint $P_{i+1} = P_{i+1}^T$. Using the facts that both P_i and X_i are symmetric and that \tilde{Z} is anti-symmetric we get, $P_{i+1}^T = P_i + \alpha(X_i - X_i P_i) - \tilde{Z}$. By solving the equation $P_{i+1} = P_{i+1}^T$ we extract the value of $\tilde{Z} = \frac{1}{2}\alpha(P_i X_i - X_i P_i)$. We finally get the update rule of the algorithm,

$$P_{i+1} = P_i + \alpha \left[X_i - \frac{1}{2} (P_i X_i + X_i P_i) \right]. \quad (6)$$

For the analysis of the algorithm we find it convenient to change variables,

$$P_{i+1} = P_i + \gamma_i \left[\hat{X}_i - \frac{1}{2} (P_i \hat{X}_i + \hat{X}_i P_i) \right], \quad \gamma_i = \alpha \|\mathbf{x}_i\|^2. \quad (7)$$

The algorithm can be viewed as performing a (stochastic) gradient descent, since the right term of Eq. (7) equal to the *symmetric* part of the gradient $\nabla_P \ell(\mathbf{x}, P \mathbf{x}_i)|_{P=P_i}$. The algorithm is summarized in Fig. 1. The description of the `Regularize` procedure is deferred to Sec. 4 and for now we ignore it. We refer to this algorithm as the GST algorithm, for Gradient-decent-based Subspace Tracker.

To conclude this section we show that our algorithm can be combined with Mercer kernels. We show that P_i can be written as a linear combination of outer product of the input points with coefficients $\Gamma_{p,q}$, that is, $P_i = \sum_{p,q=1}^{i-1} \Gamma_{p,q} \hat{\mathbf{x}}_p \hat{\mathbf{x}}_q^T$. The proof proceeds by induction. The initial matrix $P_1 = 0$ clearly can be written in the required form. For the induction step we substitute $\hat{X} = \hat{\mathbf{x}} \hat{\mathbf{x}}^T$ in Eq. (7) and use the induction assumption,

$$P_{i+1} = P_i + \gamma_i \left[\hat{\mathbf{x}}_i \hat{\mathbf{x}}_i^T - \frac{1}{2} \sum_{p=1}^{i-1} \hat{\mathbf{x}}_p \left(\sum_{q=1}^{i-1} \Gamma_{p,q} \hat{\mathbf{x}}_q^T \hat{\mathbf{x}}_i \right) \hat{\mathbf{x}}_i^T - \frac{1}{2} \sum_{q=1}^{i-1} \hat{\mathbf{x}}_i \left(\sum_{p=1}^{i-1} \Gamma_{p,q} \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_p \right) \hat{\mathbf{x}}_q^T \right].$$

The terms in the brackets are of the desired form and furthermore the matrix P_i is of the desired form due to the induction assumption. From the last equation we can recursively set the values of the matrix Γ : $\Gamma_{i,i} = \gamma_i$, $\Gamma_{q,i} = \Gamma_{i,q} = \sum_{p=1}^{i-1} \Gamma_{p,q} \hat{\mathbf{x}}_i^T \hat{\mathbf{x}}_p$ for $q = 1 \dots i-1$. We have shown that all the steps of the online algorithm depends in the input data through inner product operations and thus can replace the standard inner product with any Mercer kernel.

2.1 Analysis

Before we prove a loss bound on the performance of the algorithm, we first fulfill our promise and show that indeed the algorithm maintains a positive semidefinite linear transformation P_i . This property is somewhat surprising in light of the following observation. Standard linear algebra computation shows that the rank of the matrix $\hat{X}_i - \frac{1}{2}(P_i \hat{X}_i + \hat{X}_i P_i)$ is either one or two. In the latter case, the eigenvalues of this matrix are, $\lambda_{\pm} = \frac{1}{2} \left(1 \pm \sqrt{1 + \hat{\mathbf{x}}_i^T P_i P_i \hat{\mathbf{x}}_i - (\hat{\mathbf{x}}_i^T P_i \hat{\mathbf{x}}_i)^2} \right)$. The smaller eigenvalue

λ_- is negative since $\hat{\mathbf{x}}_i^\top P_i P_i \hat{\mathbf{x}}_i - (\hat{\mathbf{x}}_i^\top P_i \hat{\mathbf{x}}_i)^2 \geq 0$. Thus on each iteration some of the eigenvalues of P_{i+1} are potentially smaller than those of P_i . If some of the eigenvalues of P_i are zero then some of the eigenvalues of P_{i+1} can be negative [11].

Specifically, we show by induction that for any linear transformation that can be derived along the run of the algorithm $P_1 \dots P_{m+1}$ we have that $0 \preceq P_{i+1} \preceq bI$ assuming $\gamma_i \in [0, a]$, where $b = 4/(4-a)$. This requirement is easily fulfilled by setting the tradeoff parameter to be in the range $\alpha \in [0, a/R^2]$ where $R^2 = \max_i \|\mathbf{x}_i\|^2$. Since the initialization of the linear transformation is such that $0 \preceq P_1 \preceq bI$, then it suffices to show that the claim holds inductively. Finally, although the lemma is general we assume below that the learning rate is set to $\alpha = 1 = a$ and thus the upper bound on the eigenvalues is $b = 4/3$.

Lemma 1. *Let $0 < a \leq 2$ and $b = 4/(4-a) > 1$. If $\gamma_i \in [0, a]$ and $0 \preceq P_i \preceq bI$ then $0 \preceq P_{i+1} \preceq bI$.*

Proof. Since P_{i+1} is symmetric by construction it is remained to show that its eigenvalues are between zero and b . Rewriting Eq. (7) we get,

$$\begin{aligned} P_{i+1} &= P_i + \gamma_i \left[\hat{X}_i - \frac{1}{2} (P_i \hat{X}_i + \hat{X}_i P_i) \right] \\ &= \left(I - \frac{1}{2} \gamma_i \hat{X}_i \right) P_i \left(I - \frac{1}{2} \gamma_i \hat{X}_i \right) + \hat{X}_i \left(\gamma_i I - \frac{1}{4} \gamma_i^2 P_i \right) \hat{X}_i, \end{aligned} \quad (8)$$

where we used the equality $\hat{X} = \hat{X} \hat{X}$. Eq. (8) is a sum of two terms, the first term is PSD by definition and the second term is PSD as since $(1/4)\gamma_i^2 P_i \preceq (1/4)ba\gamma_i I \preceq a/(4-a)\gamma_i I \preceq \gamma_i I$. The last inequality holds since $a \leq 2$. Since PSD matrices are closed under addition we get that $0 \preceq P$. We show next that the eigenvalues of this matrix are always not greater than b and we do so by showing that for all vectors \mathbf{v} we have that $\mathbf{v}^\top P_{i+1} \mathbf{v} \leq b \|\mathbf{v}\|^2$. Using Eq. (8) we get,

$$\mathbf{v}^\top P_{i+1} \mathbf{v} = \left[\mathbf{v}^\top \left(I - \frac{1}{2} \gamma_i \hat{X}_i \right) \right] P_i \left[\left(I - \frac{1}{2} \gamma_i \hat{X}_i \right) \mathbf{v} \right] + \mathbf{v}^\top \hat{X}_i \left(\gamma_i I - \frac{1}{4} \gamma_i^2 P_i \right) \hat{X}_i \mathbf{v}.$$

We first develop the left term by computing the norm of the vector multiplying P_i ,

$$\left\| \left(I - \frac{1}{2} \gamma_i \hat{X}_i \right) \mathbf{v} \right\|^2 = \mathbf{v}^\top \left[I - \frac{1}{2} \gamma_i \hat{X}_i \right] \left[I - \frac{1}{2} \gamma_i \hat{X}_i \right] \mathbf{v} = \|\mathbf{v}\|^2 + \left(\frac{1}{4} \gamma_i^2 - \gamma_i \right) \langle \mathbf{v}, \hat{\mathbf{x}} \rangle^2.$$

Plugging into the last equation and using the assumption that the eigenvalues of P_i are not greater than b we get,

$$\begin{aligned} \mathbf{v}^\top P_{i+1} \mathbf{v} &\leq b \|\mathbf{v}\|^2 + b \left(\frac{1}{4} \gamma_i^2 - \gamma_i \right) \langle \mathbf{v}, \hat{\mathbf{x}} \rangle^2 + \langle \mathbf{v}, \hat{\mathbf{x}} \rangle^2 \left(\gamma_i - \frac{1}{4} \gamma_i^2 \hat{\mathbf{x}}_i^\top P_i \hat{\mathbf{x}}_i \right) \\ &\leq b \|\mathbf{v}\|^2 + \left(\frac{b}{4} \gamma_i^2 - (b-1)\gamma_i \right) \langle \mathbf{v}, \hat{\mathbf{x}} \rangle^2 \leq b \|\mathbf{v}\|^2, \end{aligned}$$

where the last inequality holds because $\gamma_i \leq a$. ■

We now turn and analyze the algorithm in the loss bound model. Concretely, we compare the performance of the algorithm to that of a *fixed* idempotent projection Q . The following lemma bounds the relative loss for an individual example. The proof generalizes similar bounds for vector adaptive filtering [4].

Lemma 2. *Let \mathbf{x}_i be any vector with bounded norm $\|\mathbf{x}_i\|^2 \leq R^2$. Let Q be any idempotent projection (symmetric matrix with eigenvalues either zero or one) and let the trade-off parameter be $\alpha = 1/R^2$ then,*

$$\frac{1}{2} \|P_i - Q\|^2 - \frac{1}{2} \|P_{i+1} - Q\|^2 \geq \frac{1}{2} \alpha \|P_i \mathbf{x}_i - Q \mathbf{x}_i\|^2 - \frac{1}{2} \alpha \|\mathbf{x}_i - Q \mathbf{x}_i\|^2 .$$

Before proving the lemma we like to comment that unlike relative-performance online bounds [12] for more standard problems such as classification and regression, the algorithm and the fixed projection are measured differently. The loss the algorithm suffers is measured compared to the uncorrupted point $Q \mathbf{x}_i$ and not to the input vector \mathbf{x}_i . This is because we assume that the input data were generated using Q . Therefore, the loss of the idempotent projection Q is in fact the squared norm of the noise vector.

Proof. Tediously long algebraic manipulations give,

$$\begin{aligned} \frac{1}{2} \|P_i - Q\|^2 - \frac{1}{2} \|P_{i+1} - Q\|^2 &= -\gamma_i \|\hat{\mathbf{x}}_i - Q \hat{\mathbf{x}}_i\|^2 + \gamma_i \hat{\mathbf{x}}_i^\top \left[P_i - \frac{1}{2} (Q P_i + P_i Q) \right] \hat{\mathbf{x}}_i \\ &\quad - \frac{1}{4} \gamma_i^2 \left[\|P_i \hat{\mathbf{x}}_i - \hat{\mathbf{x}}_i\|^2 + (1 - \hat{\mathbf{x}}_i^\top P_i \hat{\mathbf{x}}_i)^2 \right] + \gamma_i \|\hat{\mathbf{x}}_i - P_i \hat{\mathbf{x}}_i\|^2 \end{aligned} \quad (9)$$

Applying Cauchy-Schwartz inequality with the vectors $(I - P_i) \hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}_i$ we get

$$(1 - \hat{\mathbf{x}}_i^\top P_i \hat{\mathbf{x}}_i)^2 \leq \left\| \hat{\mathbf{x}}_i^\top - P_i \hat{\mathbf{x}}_i \right\|^2 . \quad (10)$$

Observing that the assumption $\alpha = 1/R^2$ implies $\gamma_i \leq 1$, which in turn yields $\gamma_i - \gamma_i^2/2 \geq \gamma_i/2$. We get,

$$\begin{aligned} \frac{1}{2} \|P_i - Q\|^2 - \frac{1}{2} \|P_{i+1} - Q\|^2 &\geq \frac{1}{2} \alpha \|P_i \mathbf{x}_i - \mathbf{x}_i\|^2 - \alpha \|\mathbf{x}_i - Q \mathbf{x}_i\|^2 \\ &\quad + \alpha \mathbf{x}_i^\top \left[P_i - \frac{1}{2} (Q P_i + P_i Q) \right] \mathbf{x}_i , \end{aligned} \quad (11)$$

We further derive the first term and use the fact that $\mathbf{x}^\top Q \mathbf{x} = \mathbf{x}^\top Q Q \mathbf{x}$ and get,

$$\begin{aligned} \|P_i \mathbf{x}_i - \mathbf{x}_i\|^2 &= \|P_i \mathbf{x}_i - Q \mathbf{x}_i + Q \mathbf{x}_i - \mathbf{x}_i\|^2 \\ &= \|P_i \mathbf{x}_i - Q \mathbf{x}_i\|^2 + \|Q \mathbf{x}_i - \mathbf{x}_i\|^2 - 2 \mathbf{x}_i^\top \left[P_i - \frac{1}{2} (Q P_i + P_i Q) \right] \mathbf{x}_i , \end{aligned} \quad (12)$$

Plugging Eq. (12) into Eq. (11) and rearranging the terms conclude the proof. \blacksquare

We use the Lemma 2 to prove the main result of this section.

Parameters: $\alpha > 0$; $B > 0$
Initialize: Set $P_1 = 0$
Loop: For $i = 1, 2, \dots, m$

- Get a new point $\mathbf{x}_i \in \mathbb{R}^n$
- Set $\gamma_i = \alpha \|\mathbf{x}_i\|^2$
- Update,

$$P'_{i+1} = P_i + \gamma_i \left[\hat{X}_i - \frac{1}{2} \left(P_i \hat{X}_i + \hat{X}_i P_i \right) \right]$$

- Set $P_{i+1} \leftarrow \text{Regularize}(P'_{i+1}, B)$

Output: PSD matrix $- P_{m+1}$

Parameters: $\epsilon > 0$; $B > 0$
Initialize: Set $P_1 = 0$
Loop: For $i = 1, 2, \dots, m$

- Get a new point $\mathbf{x}_i \in \mathbb{R}^n$
- Find γ_i such that Eq. (16) holds.
- Update,

$$P'_{i+1} = P_i - \frac{\gamma_i}{2 - \gamma_i} (P_i \hat{X}_i + \hat{X}_i P_i) + \gamma_i \hat{X}_i + \frac{\gamma_i^2}{2 - \gamma_i} \hat{X}_i P_i \hat{X}_i$$

- Set $P_{i+1} \leftarrow \text{Regularize}(P'_{i+1}, B)$.

Output: PSD matrix $- P_{m+1}$

Fig. 1. The GST online algorithm

Fig. 2. The PST_ϵ online algorithm

Theorem 1. Let $\mathbf{x}_1 \dots \mathbf{x}_m \dots$ be any input sequence for the PST algorithm (without the regularization). Denote by $R = \max_i \|\mathbf{x}_i\|^2$. Let Q be any idempotent projection and assume the tradeoff parameter is set to $\alpha = 1/R^2$. Then the loss the algorithm suffers is bounded as follows,

$$\sum_i \|P_i \mathbf{x}_i - Q \mathbf{x}_i\|^2 \leq \text{rank}(Q) R^2 + \sum_i \|\mathbf{x}_i - Q \mathbf{x}_i\|^2 .$$

The theorem is proved by bounding $\sum_i \|P_i - Q\|^2 - \|P_{i+1} - Q\|^2$ from above and below. For the upper bound we note that it is a telescopic sum which is less than $\text{rank}(Q)$. For the lower bound we bound each summand separately using Lemma 2. An important comment is in place. The form of the bound is identical to similar bounds for online algorithms for classification or regression [13], where the cumulative performance of the algorithm (P_i) compared to the target function (Q) is bounded by a property of the target (rank here; squared norm of a vector in classification or regression) plus the cumulative performance of a competitor compared to the target function (Q). Note that the second term in the bound is $\|I \mathbf{x}_i - Q \mathbf{x}_i\|^2$ and thus the competitor is the identity matrix I . However, there is one crucial difference. In classification and regression the target function is fixed (through the supervision) and we are free to choose any competitor. Here, the competitor is fixed (I) and we are free to choose any target (Q), which represents an arbitrary subspace underlying the data.

Intuitively, the fixed term (rank) of the bound is related to a transient period of the algorithm when it shifts from its initial subspace toward the target subspace, and the cumulative performance of the competitor bounds the performance when eventually any new vector falls approximately in the span of the vectors already processed.

To exemplify the bound let us consider two extreme cases. First, assume that indeed all the points \mathbf{x}_i lies exactly in a linear subspace of dimension $n \ll d$. So, there exists a projection Q such that $Q \mathbf{x}_i = \mathbf{x}_i$ and the second term of the bound thus vanishes. The algorithm suffers loss which is scaled linearly with the internal dimension n and is

independent of d or the number of points m . Second, consider the case that there is no underlying linear subspace. We consider two options $Q = I$ or $Q = 0$. In the former case all the points are treated as data, again with no noise. The bound however scales like the true dimension d . In the latter case all the points are considered as noise around the origin, and the loss the algorithm suffers is bounded by the total variance.

3 Projection Based Algorithm

We now turn our attention to an alternative method for deriving online algorithms. We modify the squared loss function to ignore very small distances specified by a fixed insensitivity parameter ϵ ,

$$\ell_\epsilon(\mathbf{x}, P\mathbf{x}) = \begin{cases} 0 & \|\mathbf{x} - P\mathbf{x}\| \leq \sqrt{2\epsilon} \\ (\|\mathbf{x} - P\mathbf{x}\| - \sqrt{2\epsilon})^2 & \text{Otherwise} \end{cases} .$$

That is, if the squared loss is below some predefined tolerance level, then the value of the ϵ -insensitive loss is zero. Otherwise it is equal to a shift of the squared loss. The update rule for the new algorithm sets the new matrix P_{i+1} to be the solution to the following projection problem [13],

$$\min_P \frac{1}{2} \|P - P_i\|^2 \quad \text{s.t.} \quad \ell_\epsilon(\mathbf{x}_i, P\mathbf{x}_i) = 0, P = P^T, P \succeq 0 .$$

The solution of the optimization problem is the projection of P onto the intersection of the positive semidefinite cone and the second order body of matrices P that satisfy $\|\mathbf{x}_i - P\mathbf{x}_i\| \leq \epsilon$ and are centered at the identity matrix I . Clearly the subset of matrices defined by the intersection is not empty as it contains the identity matrix. We refer to this algorithm as the PST_ϵ algorithm, for Projection based algorithm for Subspace Tracking with insensitivity level ϵ .

As with the GST algorithm, we derive an update rule by computing the corresponding Lagrangian. As before we omit for now the constraint of being positive semidefinite. We show below that the PSD constraint is indeed satisfied by the optimal solution.

$$\mathcal{L}(P; \alpha_i) = \frac{1}{2} \|P - P_i\|^2 + \alpha_i \left[\frac{1}{2} \|\mathbf{x}_i - P\mathbf{x}_i\|^2 - \epsilon \right] - \text{Tr} [Z(P - P^T)] , \quad (13)$$

where $\alpha_i \geq 0$ is the Lagrange multiplier. To solve the problem we first differentiate \mathcal{L} with respect to P and set the result to zero, $P_{i+1}(I + \alpha_i X_i) = P_i + \alpha_i X_i + \tilde{Z}$, where $\tilde{Z} = Z^T - Z$ is an anti-symmetric matrix. We solve the last equation by first computing the inverse of the matrix $I + \alpha_i X_i$ and then solving for \tilde{Z} . The details are omitted for lack of space. As before we define additional notation γ_i ,

$$\gamma_i = \frac{\alpha_i \|\mathbf{x}_i\|^2}{1 + \alpha_i \|\mathbf{x}_i\|^2} \quad , \quad \alpha_i = \frac{1}{\|\mathbf{x}_i\|^2} \frac{\gamma_i}{1 - \gamma_i} \quad (14)$$

which we use to write the update rule of this algorithm,

$$P_{i+1} = P_i + \gamma_i \hat{X}_i - \frac{\gamma_i}{2 - \gamma_i} (P_i \hat{X}_i + \hat{X}_i P_i) + \frac{\gamma_i^2}{2 - \gamma_i} \hat{X}_i P_i \hat{X}_i . \quad (15)$$

Note that by definition $\gamma_i \in [0, 1]$. The update rule still depends on the unknown α_i (or γ_i). To find the value of γ_i we use the KKT conditions. Whenever γ_i is positive the inequality constraint $\frac{1}{2}\|\mathbf{x}_i - P_i\mathbf{x}_i\|^2 \leq \epsilon$ is satisfied as equality. Long algebraic manipulations yield that the left side of this equality constraint is given by the function,

$$f(\gamma) = \frac{(1-\gamma)^2}{2(2-\gamma)^2} \|\mathbf{x}_i\|^2 \left[4\|\hat{\mathbf{x}}_i - P_i\hat{\mathbf{x}}_i\|^2 + (-4\gamma + \gamma^2) \left(1 - \hat{\mathbf{x}}_i^\top P_i \hat{\mathbf{x}}_i\right)^2 \right]. \quad (16)$$

Theoretically, we can solve analytically the equation $f(\gamma_i) = \epsilon$ since it is a degree four polynomial. In practice, we use the following lemma, which states that the function $f(\gamma)$ is monotone. By definition $\gamma_i \in [0, 1]$ and thus we can find a value of γ which is far of the exact solution by at most δ in time complexity of $-\log_2(\delta)$.

Lemma 3. *The function $f(\gamma)$ defined in Eq. (16) is monotone decreasing in γ .*

The proof is omitted due to lack of space. To summarize the description of the algorithm: after receiving \mathbf{x}_i the algorithm checks whether the Euclidean distance between \mathbf{x}_i and $P_i\mathbf{x}_i$ is below the predefined threshold, $\frac{1}{2}\|\mathbf{x}_i - P_i\mathbf{x}_i\|^2 \leq \epsilon$. If so, it does nothing. Otherwise, it performs binary search in the range $[0, 1]$ and finds a value γ_i that solve the function $f(\gamma_i) = \epsilon$. We initialize $P_1 = 0$. A sketch of the algorithm is shown in Fig. 2. To conclude, we note that the PST algorithm may be extended with Mercer kernels. The proof and construction are similar to the those of the GST algorithm.

3.1 Analysis

As in the GST algorithm, in each iteration we set P_{i+1} to be a sum of the previous matrix P_i and another matrix, as given in Eq. (15). As before, this matrix is either of degree one or two, and in the latter case one of its eigenvalues is negative.

In the following we derive the analogous of Lemma. 1, which state that the eigenvalues of each of the transformations derived along the run of the algorithm $P_1 \dots P_{m+1}$ falls in the interval $[0, 1]$, so P_i are close to be idempotent projections. This situation is simpler than for the GST algorithm, in which for all allowed learning rates the upper bound on the eigenvalues b was strictly greater than one. The proof is similar to the proof of Lemma. 1.

Lemma 4. *Throughout the running of the algorithm $0 \preceq P_i \preceq I$.*

We turn to analyze the algorithm in the loss bound model. For this algorithm we change slightly both the assumptions and the bound. Here we compare the performance of the algorithm to an idempotent projection Q with point-wise bounded noise, that is $(1/2)\|\mathbf{x}_i - Q\mathbf{x}_i\|^2 \leq \epsilon$. The corresponding loss which we bound is the epsilon insensitive version of the Euclidean loss function. Before proving the theorem we prove the following auxiliary lemma, which provides a lower bound and an upper on the optimal value of γ_i .

Lemma 5. *Let γ_i be the solution of the equality $f(\gamma_i) = \epsilon$. If $\ell_\epsilon(\mathbf{x}_i, P_i\mathbf{x}_i) > 0$ then,*

$$1 - \frac{\sqrt{2\epsilon}}{\|\mathbf{x}_i - P_i\mathbf{x}_i\|} \leq \gamma_i \leq 1 - \frac{1}{2} \frac{\sqrt{2\epsilon}}{\|\mathbf{x}_i - P_i\mathbf{x}_i\|}.$$

Proof. If $\ell_\epsilon(\mathbf{x}_i, P_i \mathbf{x}_i) > 0$ we know that γ_i is defined to be the solution of the equation $f(\gamma_i) = \epsilon$. We start with the left hand-side of the desired inequality and lower bound the second term in Eq. (16). Since $\gamma_i \in [0, 1]$ we have that $-4\gamma_i + \gamma_i^2 \leq 0$. Substituting Eq. (10) we get,

$$\begin{aligned} \epsilon &\geq \frac{(1 - \gamma_i)^2}{2(2 - \gamma_i)^2} \|\mathbf{x}_i\|^2 \left[4 \|\hat{\mathbf{x}}_i - P_i \hat{\mathbf{x}}_i\|^2 + (-4\gamma_i + \gamma_i^2) \|\hat{\mathbf{x}}_i - P_i \hat{\mathbf{x}}_i\|^2 \right] \\ &= (1 - \gamma_i)^2 \frac{1}{2} \|\mathbf{x}_i - P_i \mathbf{x}_i\|^2 . \end{aligned}$$

Solving for γ_i leads to the desired bound. For the right hand-side of the inequality we return to Eq. (16) and upper bound the right term by zero. We get,

$$\epsilon \leq \frac{(1 - \gamma_i)^2}{2(2 - \gamma_i)^2} \|\mathbf{x}_i\|^2 \left[4 \|\hat{\mathbf{x}}_i - P_i \hat{\mathbf{x}}_i\|^2 \right] \leq (1 - \gamma_i)^2 4 \frac{1}{2} \|\mathbf{x}_i - P_i \mathbf{x}_i\|^2 .$$

Solving for γ_i leads to the desired bound. ■

We are now ready to prove the main theorem of the section,

Theorem 2. *Let $\mathbf{x}_1 \dots \mathbf{x}_i \dots$ be a sequence of points. Assume that there exists an idempotent projection Q that suffers zero loss $\ell_\epsilon(\mathbf{x}_i, Q\mathbf{x}_i) = 0$ for all i . Denote by $R = \max_i \|\mathbf{x}_i\|$. Then the following bound holds for the PST algorithm (without the regularization),*

$$\sum \ell_{4\epsilon}(Q\mathbf{x}_i, P_i \mathbf{x}_i) \leq 2 \text{rank}(Q) R^2 .$$

Proof. Let P_i be the projection matrix before receiving the i th vector \mathbf{x}_i . Define $\Delta_i = \|P_i - Q\|^2 - \|P_{i+1} - Q\|^2$. We prove the theorem by bounding $\sum_{i=1}^m \Delta_i$ from above and below. First note that $\sum_{i=1}^m \Delta_i$ is a telescopic sum and therefore,

$$\sum_{i=1}^m \Delta_i = \sum_i \|P_i - Q\|^2 - \|P_{i+1} - Q\|^2 \leq \|Q\|^2 = \text{rank}(Q) . \quad (17)$$

This provides an upper bound on $\sum_i \Delta_i$. To provide a lower bound on Δ_i we apply Thm. 2.4.1 in [14] and get that $\Delta_i = \|P_i - Q\|^2 - \|P_{i+1} - Q\|^2 \geq \|P_i - P_{i+1}\|^2$. We consider two cases. If $\|Q\mathbf{x}_i - P_i \mathbf{x}_i\| \leq 2\sqrt{2}\epsilon$ we use the trivial bound $\|P_i - P_{i+1}\|^2 \geq 0 = \ell_{2\epsilon}(Q\mathbf{x}_i, P_i \mathbf{x}_i)$. Otherwise, we assume that $\|Q\mathbf{x}_i - P_i \mathbf{x}_i\| \geq 2\sqrt{2}\epsilon$. Algebraic manipulations show that,

$$\Delta_i \geq \|P_i - P_{i+1}\|^2 \geq \frac{1}{2} \gamma_i^2 \|\hat{\mathbf{x}}_i - P_i \hat{\mathbf{x}}_i\|^2 . \quad (18)$$

Substituting the lower bound of Lemma 5 we get, $\Delta_i \geq \frac{1}{2} (\|\mathbf{x}_i - P_i \mathbf{x}_i\| - \sqrt{2}\epsilon)^2 \frac{1}{\|\mathbf{x}_i\|^2}$. Using the triangle inequality we get, $\|\mathbf{x}_i - P_i \mathbf{x}_i\| \geq \|Q\mathbf{x}_i - P_i \mathbf{x}_i\| - \|Q\mathbf{x}_i - \mathbf{x}_i\| \geq \|Q\mathbf{x}_i - P_i \mathbf{x}_i\| - \sqrt{2}\epsilon$, where the last inequality holds since $\|Q\mathbf{x}_i - \mathbf{x}_i\| \leq \sqrt{2}\epsilon$. Substituting back in the last equation we get, $\Delta_i \geq \frac{1}{2} (\|Q\mathbf{x}_i - P_i \mathbf{x}_i\| - 2\sqrt{2}\epsilon)^2 \frac{1}{\|\mathbf{x}_i\|^2} \geq \frac{1}{2} \ell_{4\epsilon}(Q\mathbf{x}_i, P_i \mathbf{x}_i) / R^2$. Combining Eq. (17) together with the last equation concludes the proof. ■

The theorem tells us that if the squared norm of the noise $(1/2) \|\mathbf{x}_i - Q\mathbf{x}_i\|^2$ is bounded by ϵ , then the cumulative 4ϵ -insensitive loss the algorithm suffer is bounded. To conclude, we derived two online algorithms which reconstruct corrupted input points \mathbf{x}_i by tracking linear subspaces. The performance of both algorithms is compared to the performance of arbitrary idempotent projections Q . The learning algorithms do not know the identity of Q nor they have any feedback from Q . In this aspect the learning task is harder than typical regression or classification learning problems, as there is no supervision during the learning process.

4 Regularization

In our two algorithms, overfitting arises when the eigenvalues of the linear operator P_i have large components orthogonal to the target subspace. As a consequence, the *filtered* output $P_i\mathbf{x}_i$ will include noise components as well as the true signal. Both algorithms may suffer from this problem since in our setting there is no feedback. Therefore, our algorithms approximate the true feedback $Q\mathbf{x}_i$ using \mathbf{x}_i , which contains also noise components. Furthermore, the only goal of the update rule of both algorithms is to reduce the loss related to \mathbf{x}_i , ignoring any other issue such as the rank or trace of the transformation P_i . Therefore, as we shall see next, both algorithms favor an increase in the trace of the transformations since, in general, that reduces the loss suffered.

We exemplify our claim for the GST algorithm, similar results hold for the PST algorithm. We compute the change in the trace by using Eq. (7) and get,

$$\text{Tr}[P_{i+1}] = \text{Tr} \left[P_i + \gamma_i \left[\hat{X}_i - \frac{1}{2} (P_i \hat{X}_i + \hat{X}_i P_i) \right] \right] = \text{Tr}[P_i] + \gamma_i \hat{\mathbf{x}}_i^\top (I - P_i) \hat{\mathbf{x}}_i.$$

Examining the change in trace we observe that the single fixed point of the update is $P_i = I$, the identity matrix. Otherwise, if some of the eigenvalues of P_i are below one, the trace will increase. (Using our analogy, one-class algorithms are designed to capture the input data in a ball, a goal which favors increasing the radius of the ball.) We remind the reader that according to Lemma 1 the eigenvalues are not bounded from above by one, only by $4/3$. In this case, according to Eq. (19), the trace may slightly decrease to compensate this high value of the eigenvalues. The phenomenon is indeed observed in the simulations we performed and are shown in Sec. 5. Nevertheless, this will not stop the algorithm from overfitting, since when some of the eigenvalues are small, the update operation will increase the trace.

Following [4] we add a second step to the update rule, after the primary update of Eq. (7) (for the GST algorithm) or Eq. (15) (for the PST algorithm). Due to lack of space we focus in the GST algorithm. The algorithm employs an additional parameter $B > 0$, which is used to bound the norm of the eigenvalues after the update. We consider two versions for this update, which correspond to ℓ_1 -norm regularization and ℓ_2 -norm regularization. Intuitively, the parameter B specifies a continuous requirement that replaces a standard rank requirement. Specifically the update rule is defined as follows: we first perform the gradient step of Eq. (7) and define, $P'_{i+1} = P_i + \gamma_i \left[\hat{X}_i - \frac{1}{2} (P_i \hat{X}_i + \hat{X}_i P_i) \right]$. Then we set P_{i+1} to be the solution of the following optimization problem,

$$P_{i+1} = \arg \min_P \frac{1}{2} \|P - P'_{i+1}\|^2 \quad \text{s.t.} \quad P = P^T, P \succeq 0 \quad (19)$$

$$L = 1 \text{ (version 1)} \quad ; \quad L = 2 \text{ (version 2)} \quad \text{Tr}(P^L) \leq B \quad (20)$$

For the first version we set $L = 1$ and bound the ℓ_1 norm of the eigenvalues of P . For the second version we set $L = 2$ and bound the ℓ_2 norm of the eigenvalues of P . We now describe in detail how to solve in practice these optimization problems. Note that in both cases if the norm condition (Eq. (20)) is satisfied for P'_{i+1} then $P_{i+1} = P'_{i+1}$. We thus assume this is not the case.

Version 1 – ℓ_1 norm: We omit the derivation of the solution due to lack of space and proceed with a formal description of it. To solve the optimization problem one needs to compute the eigenvectors \mathbf{v}_j of P'_{i+1} and the corresponding eigenvalues $\lambda'_j \geq 0$. To be more concrete we write $P'_{i+1} = \sum_{j=1}^d \lambda'_j \mathbf{v}_j \mathbf{v}_j^\top$. Then the optimal solution is given by $P_{i+1} = \sum_{j=1}^d \lambda_j \mathbf{v}_j \mathbf{v}_j^\top$, where $\lambda_j = \max\{\lambda'_j - \eta, 0\}$ and η is chosen such that $\text{Tr}(P_{i+1}) = \sum_j \lambda_j = B$. Finding the value of η given the set of eigenvalues λ'_j can be computed using [15, Fig. 3] in $O(d \log d)$ time. To conclude, since it takes $O(d)$ time to compute the trace of a matrix and another $O(d^3)$ time to perform eigenvector decomposition, the time required to verify if an update is needed in $O(d)$ time and the total runtime of the update step is $O(d^3)$.

Version 2 – ℓ_2 norm: By writing the Lagrangian of the corresponding optimization problem, and taking the

derivative with respect to P we get that the solution P_{i+1} is proportional to P'_{i+1} . Using KKT conditions we can compute the constants and get the update rule for this version: if $\|P'_{i+1}\|^2 > B$ then set $P_{i+1} = P'_{i+1} \sqrt{B} / \|P'_{i+1}\|$. Otherwise, $P_{i+1} = P'_{i+1}$. To conclude, since it takes $O(d^3)$ time to multiply matrices, then it takes $O(d^3)$ time to compute $\|P'_{i+1}\|^2 = \text{Tr}(P'_{i+1} P'_{i+1})$ and verify if an update should be performed. If so, the update step takes $O(d^2)$ time since it involves a scaling of each element of P'_{i+1} .

The following theorem bounds the loss the modified GST algorithm suffers. As we shall see, although we consider a restricted set of projections P_i with a bounded trace or bounded trace squared, the performance of the algorithm does not deteriorate assuming it is compared to an idempotent projection Q under the same restriction. In other words, the loss bound for both versions has the same form. The only difference is the touchstone idempotent projection we use. Originally there were no restrictions over both the projection P_i and the reference projection Q . However, the modified algorithm restricts the

Input: PSD matrix P' ; Bound $B > 0$

Version 1:

If $\text{Tr}[P'] > B$ then

- Compute the eigen-decomposition of P' ,
 $P' = \sum_{j=1}^d \lambda'_j \mathbf{v}_j \mathbf{v}_j^\top$
- Find η such that $\sum_j \max\{\lambda'_j - \eta, 0\} = B$.
- Set $P = \sum_{j=1}^d \max\{\lambda'_j - \eta, 0\} \mathbf{v}_j \mathbf{v}_j^\top$

Else : $P = P'$.

Version 2:

- If $\text{Tr}[P'^2] > B$ then set $P = P' \frac{\sqrt{B}}{\|P'\|}$
- Else : $P = P'$.

Return: PSD matrix – P

Fig. 3. The Regularize Procedure

projection P_i the algorithm maintains, and the corresponding analysis assumes similar restriction over the reference idempotent projection Q .

Restricting the set of possible projections to have eigenvalues with bounded norm has an additional benefit. It allows the algorithm (and the analysis) to perform well even if the sequence of input vectors is *not stationary*. Specifically, we no longer compare the algorithm to the performance of a *single* fixed idempotent projection Q , which corresponds to a fixed subspace. We allow more complicated comparisons in which different segments of the input points may be best filtered with a unique own idempotent projection.

Theorem 3. *Let $\mathbf{x}_1 \dots \mathbf{x}_m$ be any input for the algorithm. Denote by $R = \max_i \|\mathbf{x}_i\|^2$. Let Q_i (for $i = 1 \dots m$) be any sequence of idempotent projections $Q_i^2 = Q_i$ with bounded trace $\text{Tr}[Q_i] \leq B$. Assume the tradeoff parameter is set to $\alpha = 1/R^2$. Then,*

$$\sum_i \|P_i \mathbf{x}_i - Q_i \mathbf{x}_i\|^2 \leq R^2 B + \sum_i \|\mathbf{x}_i - Q_i \mathbf{x}_i\|^2 + A \quad , \text{ where ,}$$

$$A = \sum_i B \|Q_i - Q_{i+1}\|_\infty \quad (\mathbf{ver 1}) \quad A = \sum_i \sqrt{B} \|Q_i - Q_{i+1}\|_2 \quad (\mathbf{ver 2})$$

As Theorem 1, the bound includes a fixed penalty term and the cumulative loss suffered by a series of projection functions. For the case of non-stationary data it contains an additional penalty term for deviation of these projections. The skeleton of the proof is similar to the proof of analogous theorem in [4], but it is more involved since we are dealing with PSD matrices and not vectors.

5 Simulations

The theory developed so far can be nicely illustrated via some simple simulations. We briefly describe two such experiments. In the first experiment we generated 3, 000 points in \mathbb{R}^2 . All the points lie in a linear subspace of degree 1 and in the unit ball around the origin. We added random uniform noise with maximal value of 0.1. We ran the GST algorithm with no regularization, using ℓ_1 regularization and using ℓ_2 regularization. In the latter cases we set $B = 1$ - the true dimension. The plot in the top-leftmost panel of Fig. 4 shows the cumulative squared error relative to the clean data. That is, the value at location j is $\sum_i^j \|P_i \mathbf{x}_i - Q \mathbf{x}_i\|^2$. Empirically, without regularization the performance is about four time worse than using regularization. Furthermore, the ℓ_1 regularization performs better than the ℓ_2 regularization. An explanation of these results appear in the top second-left panel. For each of the three algorithms we applied the projection obtained in the end of the training process P_{3001} on the unit circle and generating an ellipsoid. The axes of the ellipsoid correspond to the directions of the eigenvectors, and their relative length correspond to the eigenvalues. The plot also shows the same transformation of the unit circle using the competitor Q , which is represented as a black-solid line. (This is because it is of rank 1.) From the plot we observe that without regularization, the matrix P_{3001} is essentially the identity matrix, and no filtering is performed. The second eigenvalue of the matrix P_{3001} when using the ℓ_1 regularization is much smaller than the second eigenvalue when using the ℓ_2 regularization. This is reflected in the fact that one ellipsoid is more skewed than the other. Note that although the rank of the

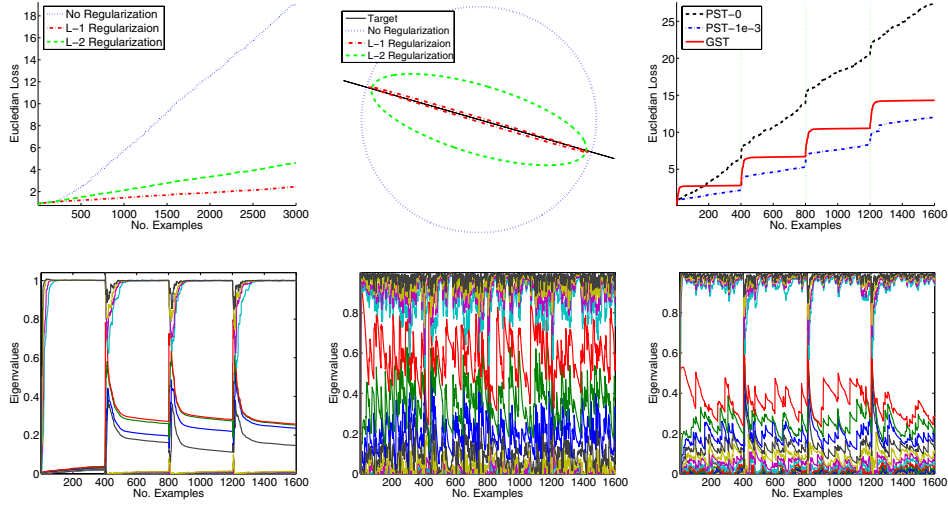


Fig. 4. Top, left to right: cumulative sum of ℓ_2 discrepancy evaluated with clean data for the first simulation and illustration of projection matrices obtained after training. Top-right : cumulative sum of ℓ_2 discrepancy for the second simulation. Bottom, left to right: top 20 eigenvalues of the projection matrix P for GST , PST_0 and PST_{1e-3} .

ℓ_1 matrix P_{3001} is closer to be one, the major subspace (which corresponds to the larger eigenvalue) is similar, but not identical, to the true subspace. This relationship between ℓ_1 and ℓ_2 regularization (where the former generates sparser solutions) appears in other contexts in machine learning. Our case is unique since here the ℓ_1 regularization generates a matrix with sparse eigen-spectrum and not a sparse matrix.

In the second experiment we repeated the following process four times. We picked at random 400 points in \mathbb{R}^{80} . All the points lie in a linear subspace of degree 4 and in the unit ball around the origin. We added random uniform noise with maximal value of 0.1. Finally, we concatenated the four sequences into a single sequence of length 1,600. We run three algorithms: GST , PST_0 and PST_{1e-3} . We ran all algorithms with ℓ_2 regularization and set $B = 5$, the actual dimension. The top-right panel of Fig. 4 shows the cumulative squared error relative to the clean data. The PST_0 algorithm performs worst, the GST algorithm second, and the PST_{1e-3} algorithm is the best. One possible explanation is that for $\epsilon = 0$ the PST tracks the noise since by definition $P_{i+1}x_i = x_i$. The two other algorithms cope with noise in different way, either by using a sensible learning rate or using the predefined tolerance ϵ . Interestingly, as indicated by the “stair-shaped” graph, the GST algorithm is more sensitive to the shift between chunks compared to being inside chunks, and vice versa for the PST algorithm. In other words, if we know that the subspaces will not be shifted or switched frequently, then GST is better, while PST is better to track non-stationary subspaces.

The three plots in the bottom of Fig. 4 show the top 20 eigenvalues of P_i for GST , PST_0 and PST_{1e-3} (left to right) at each time step. The eigenvalues of GST are smooth as a function of time. Note, as suggested by Lemma. 1, some eigenvalues of P_i are indeed larger than unit. For PST_0 , although the level of eigenvalues corresponding to the true subspace is constantly higher than the eigenvalues level for the noise, the

gap is small. Again, it seems because this algorithm is fitting the noise and is adopting non-relevant directions. Finally, PST_{1e-3} shows noisier behavior compared with GST.

To conclude the paper we presented and analyzed two algorithm for online subspace tracking and two regularization schemas. The simulations performed demonstrate the merits of our approach. There are many possible extensions for this work. An interesting question is extending this algorithmic framework to track affine subspaces and not only the special case of linear subspaces. The relation between linear subspaces and affine subspaces is similar to the relation between linear classifiers through the origin and general linear classifiers. Another interesting direction is to design batch algorithms for PCA which optimize loss functions other than the traditional Euclidean distance. A possible approach is to write a global SDP similar to the one solved in the PST algorithm. A viable research direction is to use low-rank regularization instead of the low-norm regularization used in this paper. This may lead to more efficient representation and faster algorithms. Finally, it seems that there are many similarities between adaptive signal processing and online learning. This paper explore one such relation.

Acknowledgements. The author thanks John Blitzer, Dean Foster, Dan Lee, Fernando Pereira, Lawrence Saul and Abraham Weiner for many fruitful discussions.

References

1. J.P. Delmas and J.F. Cardoso. Performance analysis of an adaptive algorithm for tracking dominant subspaces. *IEEE Transactions on Signal Processing*, 46:3054–3057, 1998.
2. R. Kumaresan and D.W. Tufts. Estimating the angles of arrival of multiple plane waves. *IEEE Transactions on Aerospace and Electronic Systems*, 19:134–139, 1983.
3. A. M. Haimovich and Y. Bar-Ness. An eigenanalysis interference canceler. *IEEE Transactions on Signal Processing*, 39:76–84, 1991.
4. J. Kivinen, M. K. Warmuth, and B. Hassibi. The p-norm generalization of the lms algorithm for adaptive filtering. In *Proc. 13th IFAC Symposium on System Identification*, 2003.
5. A.H. Sayed. *Fundamentals of Adaptive Filtering*. Wiley-Interscience, 2003.
6. D.M.J. Tax. *One-class classification; Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology, 2001.
7. H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis, 2004.
8. K. Q. Weinberger, John C. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS 18*, 2005.
9. S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In *Proc. of the 21st international conf. on Machine learning*, page 94. ACM Press, 2004.
10. k. Tsuda, G. Rätsch, and M.K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.
11. G.H. Golub and C.F. Van Loan. *Matrix computations*. John Hopkins University Press, 1989.
12. J. Kivinen, D.P. Helmbold, and M. Warmuth. Relative loss bounds for single neurons. *IEEE Transactions on Neural Networks*, 10(6):1291–1304, 1999.
13. K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. In *Advances in Neural Information Processing Systems 16*, 2003.
14. Y. Censor and S.A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press, New York, NY, USA, 1997.
15. K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47, 2002.