Journal of Machine Learning Research 12 (2011) 593-625 Submitted 6/10; Revised 9/10; Published 2/11

Regression on Fixed-Rank Positive Semidefinite Matrices: A Riemannian Approach

Gilles Meyer G.MEYER@ULG.AC.BE
Department of Electrical Engineering and Computer Science
University of Liège
B-4000 Liège, Belgium

Silvère Bonnabel SILVERE.BONNABEL@MINES-PARISTECH.FR
Robotics center
Mines ParisTech
Boulevard Saint-Michel, 60, 75272 Paris, France

Rodolphe Sepulchre R.SEPULCHRE@ULG.AC.BE
Department of Electrical Engineering and Computer Science
University of Liège
B-4000 Liège, Belgium

Editor: Inderjit Dhillon

Abstract

The paper addresses the problem of learning a regression model parameterized by a fixed-rank positive semidefinite matrix. The focus is on the nonlinear nature of the search space and on scalability to high-dimensional problems. The mathematical developments rely on the theory of gradient descent algorithms adapted to the Riemannian geometry that underlies the set of fixed-rank positive semidefinite matrices. In contrast with previous contributions in the literature, no restrictions are imposed on the range space of the learned matrix. The resulting algorithms maintain a linear complexity in the problem size and enjoy important invariance properties. We apply the proposed algorithms to the problem of learning a distance function parameterized by a positive semidefinite matrix. Good performance is observed on classical benchmarks.

Keywords: linear regression, positive semidefinite matrices, low-rank approximation, Riemannian geometry, gradient-based learning

1. Introduction

A fundamental problem of machine learning is the learning of a distance between data samples. When the distance can be written as a quadratic form (either in the data space (Mahalanobis distance) or in a kernel feature space (kernel distance)), the learning problem is a regression problem on the set of positive definite matrices. The regression problem is turned into the minimization of the prediction error, leading to an optimization framework and gradient-based algorithms.

The present paper focuses on the nonlinear nature of the search space. The classical framework of gradient-based learning can be generalized provided that the nonlinear search space is equipped with a proper Riemannian geometry. Adopting this general framework, we design novel learning algorithms on the space of fixed-rank positive semidefinite matrices, denoted by $S_+(r,d)$, where d

is the dimension of the matrix, and r is its rank. Learning a parametric model in $S_+(r,d)$ amounts to jointly learn a r-dimensional subspace and a quadratic distance in this subspace.

The framework is motivated by *low-rank learning* in large-scale applications. If the data space is of dimension d, the goal is to maintain a linear computational complexity $O(d)$. In contrast to the classical approach of first reducing the dimension of the data and then learning a distance in the reduced space, there is an obvious conceptual advantage to perform the two tasks simultaneously. If this objective can be achieved without increasing the numerical cost of the algorithm, the advantage becomes also practical.

Our approach makes use of two quotient geometries of the set $S_+(r,d)$ that have been recently studied by Journée et al. (2010) and Bonnabel and Sepulchre (2009). Making use of a general theory of line-search algorithms in quotient matrix spaces (Absil et al., 2008), we obtain concrete gradient updates that maintain the rank and the positivity of the learned model at each iteration. This is because the update is intrinsically constrained to belong to the nonlinear search space, in contrast to early learning algorithms that neglect the non linear nature of the search space in the update and impose the constraints a posteriori (Xing et al., 2002; Globerson and Roweis, 2005).

Not surprisingly, our approach has close connections with a number of recent contributions on learning algorithms. Learning problems over nonlinear matrix spaces include the learning of subspaces (Crammer, 2006; Warmuth, 2007), rotation matrices (Arora, 2009), and positive definite matrices (Tsuda et al., 2005). The space of (full-rank) positive definite matrices $S_+(d)$ is of particular interest since it coincides with our set of interest in the particular case $r = d$.

The use of Bregman divergences and alternating projection has been recently investigated for learning in $S_+(d)$. Tsuda et al. (2005) propose to use the *von Neumann* divergence, resulting in a generalization of the well-known AdaBoost algorithm (Schapire and Singer, 1999) to positive definite matrices. The use of the so-called *LogDet* divergence has also been investigated by Davis et al. (2007) in the context of Mahalanobis distance learning.

More recently, algorithmic work has focused on scalability in terms of dimensionality and data set size. A natural extension of the previous work on positive definite matrices is thus to consider low-rank positive semidefinite matrices. Indeed, whereas algorithms based on full-rank matrices scale as $O(d^3)$ and require $O(d^2)$ storage units, algorithms based on low-rank matrices scale as $O(dr^2)$ and require $O(dr)$ storage units (Fine et al., 2001; Bach and Jordan, 2005). This is a significant complexity reduction as the approximation rank r is typically very small compared to the dimension of the problem d.

Extending the work of Tsuda et al. (2005), Kulis et al. (2009) recently considered the learning of positive semidefinite matrices. The authors consider Bregman divergence measures that enjoy convexity properties and lead to updates that preserve the rank as well as the positive semidefinite property. However, these divergence-based algorithms intrinsically constrain the learning algorithm to a fixed range space. A practical limitation of this approach is that the subspace of the learned matrix is fixed beforehand by the initial condition of the algorithm.

The approach proposed in the present paper is in a sense more classical (we just perform a line-search in a Riemannian manifold) but we show how to interpret Bregman divergence based algorithms in our framework. This is potentially a contribution of independent interest since a general convergence theory exists for line-search algorithms on Riemannian manifolds. The generality of the proposed framework is of course motivated by the non-convex nature of the rank constraint.

The paper is organized as follows. Section 2 presents the general optimization framework of Riemannian learning. This framework is then applied to the learning of subspaces (Section 4), positive definite matrices (Section 5) and fixed-rank positive semidefinite matrices (Section 6). The novel proposed algorithms are presented in Section 7. Section 8 discusses the relationship to existing work as well as extensions of the proposed approach. Applications are presented in Section 9 and experimental results are presented in Section 10.

2. Linear Regression on Riemannian Spaces

We consider the following standard regression problem. Given

(i) data points $\mathbf{X}$, in a linear data space $\mathcal{X} = \mathbb{R}^{d\times d}$,

(ii) observations y, in a linear output space $\mathcal{Y} = \mathbb{R}$, (or $\mathbb{R}^d$),

(iii) a regression model $\hat{y} = \hat{y}_{\mathbf{W}}(\mathbf{X})$ parameterized by a matrix $\mathbf{W}$ in a search space $\mathcal{W}$,

(iv) a quadratic loss function $\ell(\hat{y},y) = \frac{1}{2}(\hat{y}-y)^2$,

find the optimal fit $\mathbf{W}^*$ that minimizes the *expected cost*

$$F(\mathbf{W}) = \mathbb{E}_{\mathbf{X},y}\{\ell(\hat{y},y)\} = \int \ell(\hat{y},y)\, dP(\mathbf{X},y),$$

where $\ell(\hat{y},y)$ penalizes the discrepancy between observations and predictions, and $P(\mathbf{X},y)$ is the (unknown) joint probability distribution over data and observation pairs. Although our main interest will be in the scalar model

$$\hat{y} = \mathrm{Tr}(\mathbf{W}\mathbf{X}),$$

the theory applies equally to vector data points $\mathbf{x} \in \mathbb{R}^d$, $\hat{y} = \mathrm{Tr}(\mathbf{W}\mathbf{x}\mathbf{x}^T) = \mathbf{x}^T\mathbf{W}\mathbf{x}$, to a regression model parameterized by a vector $\mathbf{w} \in \mathbb{R}^d$, $\hat{y} = \mathbf{w}^T\mathbf{x}$, or to a vector output space $\hat{y} = \mathbf{W}\mathbf{x}$.

As it is generally not possible to compute $F(\mathbf{W})$ explicitly, batch learning algorithms minimize instead the *empirical cost*

$$f_n(\mathbf{W}) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \tag{1}$$

which is the average loss computed over a finite number of samples $\{(\mathbf{X}_i, y_i)\}_{i=1}^n$.

Online learning algorithms (Bottou, 2004) consider possibly infinite sets of samples $\{(\mathbf{X}_t, y_t)\}_{t\geq 1}$, received one at a time. At time t, the online learning algorithm minimizes the instantaneous cost

$$f_t(\mathbf{W}) = \frac{1}{2}(\hat{y}_t - y_t)^2.$$

In the sequel, we only present online versions of algorithms to shorten the exposition. The single necessary change to convert an online algorithm into its batch counterpart is to perform, at each iteration, the minimization of the empirical cost f_n instead of the minimization of the instantaneous cost f_t. In the sequel, we denote by f the cost function that is minimized at each iteration.

Our focus will be on *nonlinear* search spaces $\mathcal{W}$. We only require $\mathcal{W}$ to have the structure of a Riemannian matrix manifold. Following Absil et al. (2008), an abstract gradient descent algorithm can then be derived based on the update formula

$$\mathbf{W}_{t+1} = R_{\mathbf{W}_t}(-s_t\, \mathrm{grad} f(\mathbf{W}_t)). \tag{2}$$

The gradient $\operatorname{grad} f(\mathbf{W}_t)$ is an element of the tangent space $T_{\mathbf{W}_t}\mathcal{W}$. The scalar $s_t > 0$ is the step size. The retraction $R_{\mathbf{W}_t}$ is a mapping from the tangent space $T_{\mathbf{W}_t}\mathcal{W}$ to the Riemannian manifold. Under mild conditions on the retraction R, the classical convergence theory of line-search algorithms in linear spaces generalizes to Riemannian manifolds (see Absil et al., 2008, Chapter 4).

Observe that the standard (online) learning algorithm for linear regression in $\mathbb{R}^d$,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - s_t(\mathbf{w}_t^T\mathbf{x}_t - y_t)\mathbf{x}_t, \tag{3}$$

can be interpreted as a particular case of (2) for the linear model $\hat{y} = \mathbf{w}^T\mathbf{x}$ in the *linear* search space $\mathcal{W} = \mathbb{R}^d$. The Euclidean metric turns $\mathbb{R}^d$ in a (flat) Riemannian manifold. For a scalar function $f : \mathbb{R}^d \to \mathbb{R}$ of $\mathbf{w}$, the gradient satisfies

$$Df(\mathbf{w})[\boldsymbol{\delta}] = \boldsymbol{\delta}^T \operatorname{grad} f(\mathbf{w}),$$

where $Df(\mathbf{w})[\boldsymbol{\delta}]$ is the directional derivative of f in the direction $\boldsymbol{\delta}$, and the natural retraction

$$R_{\mathbf{w}_t}(-s_t \operatorname{grad} f(\mathbf{w}_t)) = \mathbf{w}_t - s_t \operatorname{grad} f(\mathbf{w}_t),$$

induces a line-search along “straight lines" which are geodesics (that is paths of shortest length) in linear spaces. With $f(\mathbf{w}) = \frac{1}{2}(\mathbf{w}^T\mathbf{x} - y)^2$, one arrives at (3).

This example illustrates that the main ingredients to obtain a concrete algorithm are convenient formulas for the gradient and for the retraction mapping. This paper provides such formulas for three examples of nonlinear matrix search spaces: the Grassmann manifold (Section 4), the cone of positive definite matrices (Section 5), and the set of fixed-rank positive semidefinite matrices (Section 6). Each of those sets will be equipped with *quotient Riemannian geometries* that provide convenient formulas for the gradient and for the retractions. Line-search algorithms in quotient Riemannian spaces are discussed in detail in the book of Absil et al. (2008). For the readers convenience, basic concepts and notations are introduced in the next section.

3. Line-Search Algorithms on Matrix Manifolds

This section summarizes the exposition of Absil et al. (2008, Chapters 3 and 4).

Restrictions on the search space are generally encoded into optimization algorithms by means of particular constraints or penalties expressed as a function of the search variable. However, when the search space is endowed with a particular manifold structure, it is possible to design an exploration strategy that is consistent with the geometry of the problem and that appropriately turns the problem into an unconstrained optimization problem. This approach is the purpose of optimization algorithms defined on matrix manifolds.

Informally, a manifold $\mathcal{W}$ is a space endowed with a differentiable structure. One usually makes the distinction between embedded submanifolds (subsets of larger manifolds) and quotient manifolds (manifolds described by a set of equivalence classes). An intuitive example of embedded submanifold is the sphere embedded in $\mathbb{R}^d$. A typical example of quotient manifold is the set of r-dimensional subspaces in $\mathbb{R}^d$, viewed as a collection of r-dimensional orthogonal frames that cannot be superposed by a rotation. The rotational variants of a given frame thus define an equivalence class (denoted using square brackets $[\cdot]$), which is identified as a single point on the quotient manifold.

To develop line-search algorithms, the notion of gradient of a scalar cost function needs to be extended to manifolds. For that purpose, the manifold $\mathcal{W}$ is endowed with a metric $g_{\mathbf{W}}(\xi_{\mathbf{W}}, \zeta_{\mathbf{W}})$,

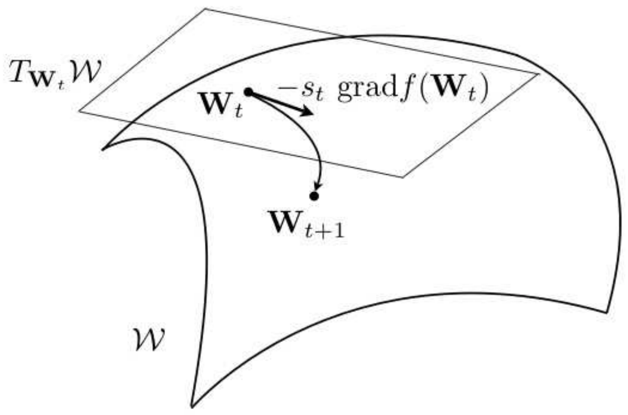

$T_{\mathbf{W}_t}\mathcal{W}$
$\mathbf{W}_t$
$-s_t\ \mathrm{grad} f(\mathbf{W}_t)$
$\mathbf{W}_{t+1}$
$\mathcal{W}$

Figure 1: Gradient iteration on a Riemannian manifold. The search direction $-\mathrm{grad}\, f(\mathbf{W}_t)$ belongs to the tangent space $T_{\mathbf{W}_t}\mathcal{W}$. The updated point $\mathbf{W}_{t+1}$ automatically remains inside the manifold thanks to the retraction mapping.

which is an inner product defined between elements $\xi_{\mathbf{W}}, \zeta_{\mathbf{W}}$ of the tangent space $T_{\mathbf{W}}\mathcal{W}$ at $\mathbf{W}$. The metric induces a norm on the tangent space $T_{\mathbf{W}}\mathcal{W}$ at $\mathbf{W}$:

$$\|\xi_{\mathbf{W}}\|_{\mathbf{W}} = \sqrt{g_{\mathbf{W}}(\xi_{\mathbf{W}}, \xi_{\mathbf{W}})}.$$

The gradient of a smooth scalar function $f : \mathcal{W} \to \mathbb{R}$ at $\mathbf{W} \in \mathcal{W}$ is the only element $\mathrm{grad}\, f(\mathbf{W}) \in T_{\mathbf{W}}\mathcal{W}$ that satisfies

$$Df(\mathbf{W})[\boldsymbol{\Delta}] = g_{\mathbf{W}}(\boldsymbol{\Delta}, \mathrm{grad}\, f(\mathbf{W})), \quad \forall \boldsymbol{\Delta} \in T_{\mathbf{W}}\mathcal{W},$$

where $\boldsymbol{\Delta}$ is a matrix representation of a “geometric" tangent vectors ξ, and where

$$Df(\mathbf{W})[\boldsymbol{\Delta}] = \lim_{t\to 0} \frac{f(\mathbf{W} + t\boldsymbol{\Delta}) - f(\mathbf{W})}{t},$$

is the standard directional derivative of f at $\mathbf{W}$ in the direction $\boldsymbol{\Delta}$.

For quotient manifolds $\mathcal{W} = \overline{\mathcal{W}}/\sim$, where $\overline{\mathcal{W}}$ is the total space and $\sim$ is the equivalence relation that defines the quotient, the tangent space $T_{[\mathbf{W}]}\mathcal{W}$ at $[\mathbf{W}]$ is sufficiently described by the directions that do not induce any displacement in the set of equivalence classes $[\mathbf{W}]$. This is achieved by restricting the tangent space at $[\mathbf{W}]$ to horizontal vectors $\bar{\xi}_{\mathbf{W}} \in T_{\mathbf{W}}\overline{\mathcal{W}}$ at $\mathbf{W}$ that are orthogonal to the equivalence class $[\mathbf{W}]$. Provided that the metric $\bar{g}_{\mathbf{W}}$ in the total space is invariant along the equivalence classes, it defines a metric in the quotient space

$$g_{[\mathbf{W}]}(\xi_{[\mathbf{W}]}, \zeta_{[\mathbf{W}]}) \triangleq \bar{g}_{\mathbf{W}}(\bar{\xi}_{\mathbf{W}}, \bar{\zeta}_{\mathbf{W}}).$$

The horizontal gradient $\overline{\mathrm{grad}\, f(\mathbf{W})}$ is obtained by projecting the gradient $\mathrm{grad}\, f(\mathbf{W})$ in the total space onto the set of horizontal vectors $\bar{\xi}_{\mathbf{W}}$ at $\mathbf{W}$.

Natural displacements at $\mathbf{W}$ in a direction $\xi_{\mathbf{W}}$ on the manifold are performed by following geodesics (paths of shortest length on the manifold) starting from $\mathbf{W}$ and tangent to $\xi_{\mathbf{W}}$. This is performed by means of the exponential mapping

$$\mathbf{W}_{t+1} = \mathrm{Exp}_{\mathbf{W}_t}(s_t \xi_{\mathbf{W}_t}),$$

which induces a line-search algorithm along geodesics.

A more general update formula is obtained if we relax the constraint of moving along geodesics. The retraction mapping

$$\mathbf{W}_{t+1} = R_{\mathbf{W}_t}(s_t \xi_{\mathbf{W}_t}),$$

locally approximates the exponential mapping. It provides an attractive alternative to the exponential mapping in the design of optimization algorithms on manifolds, as it reduces the computational complexity of the update while retaining the essential properties that ensure convergence results. When $\xi_{\mathbf{W}_t}$ coincide with $-\mathrm{grad}\, f(\mathbf{W}_t)$ a gradient descent algorithm on the manifold is obtained. Figure 1 pictures a gradient descent update on $\mathcal{W}$.

4. Linear Regression on the Grassmann Manifold

As a preparatory step to Section 6, we review the online subspace learning (Oja, 1992; Crammer, 2006; Warmuth, 2007) in the present framework. Let $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$, and consider the linear model

$$\hat{\mathbf{y}} = \mathbf{U}\mathbf{U}^T\mathbf{x},$$

with $\mathbf{U} \in \mathrm{St}(r,d) = \{\mathbf{U} \in \mathbb{R}^{d\times r} \text{ s.t. } \mathbf{U}^T\mathbf{U} = \mathbf{I}\}$, the Stiefel manifold of r-dimensional orthonormal bases in $\mathbb{R}^d$. The quadratic loss is then

$$f(\mathbf{U}) = \ell(\hat{\mathbf{y}}, \mathbf{x}) = \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{x}\|_2^2 = \frac{1}{2}\|\mathbf{U}\mathbf{U}^T\mathbf{x} - \mathbf{x}\|_2^2. \tag{4}$$

Because the cost (4) is invariant by orthogonal transformation $\mathbf{U} \mapsto \mathbf{U}\mathbf{O}$, $\mathbf{O} \in \mathcal{O}(r)$, where $\mathcal{O}(r) = \mathrm{St}(r,r)$ is the orthogonal group, the search space is in fact a set of equivalence classes

$$[\mathbf{U}] = \{\mathbf{U}\mathbf{O} \text{ s.t. } \mathbf{O} \in \mathcal{O}(r)\}.$$

This set is denoted by $\mathrm{St}(r,d)/\mathcal{O}(r)$. It is a *quotient representation* of the set of r-dimensional subspaces in $\mathbb{R}^d$, that is, the Grassmann manifold $\mathrm{Gr}(r,d)$. The quotient geometries of $\mathrm{Gr}(r,d)$ have been well studied (Edelman et al., 1998; Absil et al., 2004). The metric

$$g_{[\mathbf{U}]}(\xi_{[\mathbf{U}]}, \zeta_{[\mathbf{U}]}) \triangleq \bar{g}_{\mathbf{U}}(\bar{\xi}_{\mathbf{U}}, \bar{\zeta}_{\mathbf{U}}),$$

is induced by the standard metric in $\mathbb{R}^{d\times r}$,

$$\bar{g}_{\mathbf{U}}(\mathbf{\Delta}_1, \mathbf{\Delta}_2) = \mathrm{Tr}(\mathbf{\Delta}_1^T \mathbf{\Delta}_2),$$

which is invariant along the fibers, that is, equivalence classes. Tangent vectors $\xi_{[\mathbf{U}]}$ at $[\mathbf{U}]$ are represented by horizontal tangent vectors $\bar{\xi}_{\mathbf{U}}$ at $\mathbf{U}$:

$$\bar{\xi}_{\mathbf{U}} = \Pi_{\mathbf{U}}\mathbf{\Delta} = (\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{\Delta}, \quad \mathbf{\Delta} \in \mathbb{R}^{d\times r}.$$

Therefore, the gradient admits the simple horizontal representation

$$\overline{\mathrm{grad}\, f(\mathbf{U})} = \Pi_{\mathbf{U}}\, \mathrm{grad}\, f(\mathbf{U}), \tag{5}$$

where $\operatorname{grad} f(\mathbf{U})$ is defined by the identity

$$Df(\mathbf{U})[\mathbf{\Delta}] = \bar{g}_{\mathbf{U}}(\mathbf{\Delta}, \operatorname{grad} f(\mathbf{U})).$$

A standard retraction in $\mathrm{Gr}(r,d)$ is the exponential mapping, that induces a line-search along geodesics. The exponential map has the closed-form expression

$$\mathrm{Exp}_{\mathbf{U}}(\bar{\xi}_{\mathbf{U}}) = \mathbf{U}\mathbf{V}\cos(\mathbf{\Sigma})\mathbf{V}^T + \mathbf{Z}\sin(\mathbf{\Sigma})\mathbf{V}^T, \tag{6}$$

which is obtained from a singular value decomposition of the horizontal vector $\bar{\xi}_{\mathbf{U}} = \mathbf{Z}\mathbf{\Sigma}\mathbf{V}^T$. Following Absil et al. (2004), an alternative convenient retraction in $\mathrm{Gr}(r,d)$ is given by

$$R_{\mathbf{U}}(s\bar{\xi}_{\mathbf{U}}) = [\mathbf{U} + s\bar{\xi}_{\mathbf{U}}] = \mathrm{qf}(\mathbf{U} + s\bar{\xi}_{\mathbf{U}}), \tag{7}$$

where $\mathrm{qf}(\cdot)$ is a function that extracts the orthogonal factor of the QR-decomposition of its argument. A possible advantage of the retraction (7) over the retraction (6) is that, in contrast to the SVD computation, the QR decomposition is computed in a fixed number $O(dr^2)$ of arithmetic operations.

With the formulas (5) and (7) applied to the cost function (4), the abstract update (2) becomes

$$\mathbf{U}_{t+1} = \mathrm{qf}(\mathbf{U}_t + s_t(\mathbf{I} - \mathbf{U}_t\mathbf{U}_t^T)\mathbf{x}_t\mathbf{x}_t^T\mathbf{U}_t),$$

which is Oja's update for subspace tracking (Oja, 1992).

5. Linear Regression on the Cone of Positive Definite Matrices

The learning of a full-rank positive definite matrix is recast as follows. Let $\mathcal{X} = \mathbb{R}^{d\times d}$ and $\mathcal{Y} = \mathbb{R}$, and consider the model

$$\hat{y} = \mathrm{Tr}(\mathbf{W}\mathbf{X}),$$

with $\mathbf{W} \in S_+(d) = \{\mathbf{W} \in \mathbb{R}^{d\times d} \text{ s.t. } \mathbf{W} = \mathbf{W}^T \succ 0\}$. Since $\mathbf{W}$ is symmetric, only the symmetric part of $\mathbf{X}$ will contribute to the trace. The previous model is thus equivalent to

$$\hat{y} = \mathrm{Tr}(\mathbf{W}\mathrm{Sym}(\mathbf{X})),$$

where $\mathrm{Sym}(\cdot)$ extract the symmetric part of its argument, that is, $\mathrm{Sym}(\mathbf{B}) = (\mathbf{B}^T + \mathbf{B})/2$. The quadratic loss is

$$f(\mathbf{W}) = \ell(\hat{y}, y) = \frac{1}{2}(\mathrm{Tr}(\mathbf{W}\mathrm{Sym}(\mathbf{X})) - y)^2.$$

The quotient geometries of $S_+(d)$ are rooted in the matrix factorization

$$\mathbf{W} = \mathbf{G}\mathbf{G}^T, \quad \mathbf{G} \in \mathrm{GL}(d),$$

where $\mathrm{GL}(d)$ is the set of all invertible $d \times d$ matrices. Because the factorization is invariant by rotation, $\mathbf{G} \mapsto \mathbf{G}\mathbf{O}$, $\mathbf{O} \in \mathcal{O}(d)$, the search space is once again identified to the quotient

$$S_+(d) \simeq \mathrm{GL}(d)/\mathcal{O}(d),$$

which represents the set of equivalence classes

$$[\mathbf{G}] = \{\mathbf{G}\mathbf{O} \text{ s.t. } \mathbf{O} \in \mathcal{O}(d)\}.$$

We will equip this quotient with two meaningful Riemannian metrics.

5.1 A Flat Metric on $S_+(d)$

The metric on the quotient $\mathrm{GL}(d)/O(d)$:

$$g_{[\mathbf{G}]}(\xi_{[\mathbf{G}]},\zeta_{[\mathbf{G}]}) \triangleq \bar{g}_{\mathbf{G}}(\bar{\xi}_{\mathbf{G}},\bar{\zeta}_{\mathbf{G}}),$$

is induced by the standard metric in $\mathbb{R}^{d\times d}$,

$$\bar{g}_{\mathbf{G}}(\mathbf{\Delta}_1,\mathbf{\Delta}_2) = \mathrm{Tr}(\mathbf{\Delta}_1^T\mathbf{\Delta}_2),$$

which is invariant by rotation along the set of equivalence classes. As a consequence, it induces a metric $g_{[\mathbf{G}]}$ on $S_+(d)$. With this geometry, a tangent vector $\xi_{[\mathbf{G}]}$ at $[\mathbf{G}]$ is represented by a horizontal tangent vector $\bar{\xi}_{\mathbf{G}}$ at $\mathbf{G}$ by

$$\bar{\xi}_{\mathbf{G}} = \mathrm{Sym}(\mathbf{\Delta})\mathbf{G}, \quad \mathbf{\Delta} \in \mathbb{R}^{d\times d}.$$

The horizontal gradient of

$$f(\mathbf{G}) = \ell(\hat{y},y) = \frac{1}{2}(\mathrm{Tr}(\mathbf{G}\mathbf{G}^T\mathrm{Sym}(\mathbf{X}))-y)^2, \tag{8}$$

is the unique horizontal vector $\overline{\mathrm{grad}\, f(\mathbf{G})}$ that satisfies

$$Df(\mathbf{G})[\mathbf{\Delta}] = \bar{g}_{\mathbf{G}}(\mathbf{\Delta},\overline{\mathrm{grad}\, f(\mathbf{G})}).$$

Elementary computations yield

$$\overline{\mathrm{grad}\, f(\mathbf{G})} = 2(\hat{y}-y)\mathrm{Sym}(\mathbf{X})\mathbf{G}.$$

Since the metric is flat, geodesics are straight lines and the exponential mapping is

$$\mathrm{Exp}_{\mathbf{G}}(\bar{\xi}_{\mathbf{G}}) = [\mathbf{G}+\bar{\xi}_{\mathbf{G}}] = \mathbf{G}+\bar{\xi}_{\mathbf{G}}.$$

Those formulas applied to the cost (8) turns the abstract update (2) into the simple formula

$$\mathbf{G}_{t+1} = \mathbf{G}_t - 2s_t(\hat{y}_t - y_t)\mathrm{Sym}(\mathbf{X}_t)\mathbf{G}_t, \tag{9}$$

for an online gradient algorithm and

$$\mathbf{G}_{t+1} = \mathbf{G}_t - 2s_t\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)\mathrm{Sym}(\mathbf{X}_i)\mathbf{G}_t, \tag{10}$$

for a batch gradient algorithm.

5.2 The Affine-Invariant Metric on $S_+(d)$

Because $S_+(d) \simeq \mathrm{GL}(d)/O(d)$ is the quotient of two Lie groups, its (reductive) geometric structure can be further exploited (Faraut and Koranyi, 1994). Indeed the group $\mathrm{GL}(d)$ has a natural action on $S_+(d)$ via the transformation $\mathbf{W} \mapsto \mathbf{A}\mathbf{W}\mathbf{A}^T$ for any $\mathbf{A} \in \mathrm{GL}(d)$. The affine-invariant metric admits interesting invariance properties to these transformations. To build such an affine-invariant metric, the metric at identity

$$g_{\mathbf{I}}(\xi_{\mathbf{I}},\zeta_{\mathbf{I}}) = \mathrm{Tr}(\xi_{\mathbf{I}}\zeta_{\mathbf{I}}),$$

is extended to the entire space to satisfy the invariance property

$$g_{\mathbf{I}}(\xi_{\mathbf{I}},\zeta_{\mathbf{I}}) = g_{\mathbf{W}}(\mathbf{W}^{\frac{1}{2}}\xi_{\mathbf{I}}\mathbf{W}^{\frac{1}{2}},\mathbf{W}^{\frac{1}{2}}\zeta_{\mathbf{I}}\mathbf{W}^{\frac{1}{2}}) = g_{\mathbf{W}}(\xi_{\mathbf{W}},\zeta_{\mathbf{W}}).$$

The resulting metric on $S_+(d)$ is defined by

$$g_{\mathbf{W}}(\xi_{\mathbf{W}},\zeta_{\mathbf{W}}) = \mathrm{Tr}(\xi_{\mathbf{W}}\mathbf{W}^{-1}\zeta_{\mathbf{W}}\mathbf{W}^{-1}). \tag{11}$$

The affine-invariant geometry of $S_+(d)$ has been well studied, in particular in the context of information geometry (Smith, 2005). Indeed, any positive definite matrix $\mathbf{W} \in S_+(d)$ can be identified to the multivariate normal distribution of zero mean $\mathcal{N}(0,\mathbf{W})$, whose probability density is $p(\mathbf{z};\mathbf{W}) = \frac{1}{Z}\exp(-\frac{1}{2}\mathbf{z}^T\mathbf{W}^{-1}\mathbf{z})$, where Z is a normalizing constant. Using such a metric allows to endow the space of parameters $S_+(d)$ with a distance that reflects the proximity of the probability distributions. The Riemannian metric thus distorts the Euclidean distances between positive definite matrices in order to reflect the amount of information between the two associated probability distributions. If $\xi_{\mathbf{W}}$ is a tangent vector to $\mathbf{W} \in S_+(d)$, we have the following approximation for the Kullback-Leibler divergence (up to third order terms)

$$D_{KL}(p(\mathbf{z};\mathbf{W})||p(\mathbf{z};\mathbf{W}+\xi_{\mathbf{W}})) \approx \frac{1}{2}\,g^{FIM}_{\mathbf{W}}(\xi_{\mathbf{W}},\xi_{\mathbf{W}}) = \frac{1}{2}\,g_{\mathbf{W}}(\xi_{\mathbf{W}},\xi_{\mathbf{W}}),$$

where $g^{FIM}_{\mathbf{W}}$ is the well-known Fisher information metric at $\mathbf{W}$, which coincides with the affine-invariant metric (11) (Smith, 2005). With this geometry, tangent vectors $\xi_{\mathbf{W}}$ are expressed as

$$\xi_{\mathbf{W}} = \mathbf{W}^{\frac{1}{2}}\mathrm{Sym}(\boldsymbol{\Delta})\mathbf{W}^{\frac{1}{2}}, \quad \boldsymbol{\Delta} \in \mathbb{R}^{d\times d}.$$

The gradient $\mathrm{grad}\, f(\mathbf{W})$ is given by

$$Df(\mathbf{W})[\boldsymbol{\Delta}] = g_{\mathbf{W}}(\boldsymbol{\Delta},\mathrm{grad}\, f(\mathbf{W})).$$

Applying this formula to (5) yields

$$\mathrm{grad}\, f(\mathbf{W}) = (\hat{y}-y)\mathbf{W}\mathrm{Sym}(\mathbf{X})\mathbf{W}. \tag{12}$$

The exponential mapping has the closed-form expression

$$\mathrm{Exp}_{\mathbf{W}}(\xi_{\mathbf{W}}) = \mathbf{W}^{\frac{1}{2}}\exp(\mathbf{W}^{-\frac{1}{2}}\xi_{\mathbf{W}}\mathbf{W}^{-\frac{1}{2}})\mathbf{W}^{\frac{1}{2}}. \tag{13}$$

Its first-order approximation provides the convenient retraction

$$R_{\mathbf{W}}(s\xi_{\mathbf{W}}) = \mathbf{W} - s\xi_{\mathbf{W}}. \tag{14}$$

The formulas (12) and (13) applied to the cost (5) turn the abstract update (2) into

$$\mathbf{W}_{t+1} = \mathbf{W}_t^{\frac{1}{2}}\exp(-s_t(\hat{y}_t-y_t)\mathbf{W}_t^{\frac{1}{2}}\mathrm{Sym}(\mathbf{X}_t)\mathbf{W}_t^{\frac{1}{2}})\mathbf{W}_t^{\frac{1}{2}}.$$

With the alternative retraction (14), the update becomes

$$\mathbf{W}_{t+1} = \mathbf{W}_t - s_t(\hat{y}_t-y_t)\mathbf{W}_t\mathrm{Sym}(\mathbf{X}_t)\mathbf{W}_t,$$

which is the update of Davis et al. (2007) based on the LogDet divergence (see Section 8.1).

5.3 The Log-Euclidean Metric on $S_+(d)$

For the sake of completeness, we briefly review a third Riemannian geometry of $S_+(d)$, that exploits the property

$$\mathbf{W} = \exp(\mathbf{S}), \quad \mathbf{S} = \mathbf{S}^T \in \mathbb{R}^{d\times d}.$$

The matrix exponential thus provides a global diffeomorphism between $S_+(d)$ and the linear space of $d \times d$ symmetric matrices. This geometry is studied in detail in the paper (Arsigny et al., 2007). The cost function

$$f(\mathbf{S}) = \ell(\hat{y}, y) = \frac{1}{2}(\mathrm{Tr}(\exp(\mathbf{S})\mathrm{Sym}(\mathbf{X})) - y)^2,$$

thus defines a cost function in the linear space of symmetric matrices. The gradient of this cost function is given by

$$\mathrm{grad}\, f(\mathbf{S}) = (\hat{y}_t - y_t)\mathrm{Sym}(\mathbf{X}_t),$$

and the retraction is

$$R_{\mathbf{S}}(s\xi_{\mathbf{S}}) = \exp(\log \mathbf{W} + s\xi_{\mathbf{S}}).$$

The corresponding gradient descent update is

$$\mathbf{W}_{t+1} = \exp(\log \mathbf{W}_t - s_t(\hat{y}_t - y_t)\mathrm{Sym}(\mathbf{X}_t)),$$

which is the update of Tsuda et al. (2005) based on the von Neumann divergence.

6. Linear Regression on Fixed-Rank Positive Semidefinite Matrices

We now present the proposed generalizations to fixed-rank positive semidefinite matrices.

6.1 Linear Regression with a Flat Geometry

The generalization of the results of Section 5.1 to the set $S_+(r,d)$ is a straightforward consequence of the factorization

$$\mathbf{W} = \mathbf{G}\mathbf{G}^T, \quad \mathbf{G} \in \mathbb{R}_*^{d\times r},$$

where $\mathbb{R}_*^{d\times r} = \{\mathbf{G} \in \mathbb{R}^{d\times r} \text{ s.t. } \det(\mathbf{G}^T\mathbf{G}) \neq 0\}$. Indeed, the flat quotient geometry of the manifold $S_+(d) \simeq \mathrm{GL}(d)/O(d)$ is generalized to the quotient geometry of $S_+(r,d) \simeq \mathbb{R}_*^{d\times r}/O(r)$ by a mere adaptation of matrix dimension, leading to the updates (9) and (10) for matrices $\mathbf{G}_t \in \mathbb{R}_*^{d\times r}$. The mathematical derivation of these updates is a straight application of the material presented in the paper of Journée et al. (2010), where the quotient geometry of $S_+(r,d) \simeq \mathbb{R}_*^{d\times r}/O(r)$ is studied in details. In the next section, we propose an alternative geometry that jointly learns a r-dimensional subspace and a full-rank quadratic model in this subspace.

6.2 Linear Regression with a Polar Geometry

In contrast to the flat geometry, the affine-invariant geometry of $S_+(d) \simeq \mathrm{GL}(d)/O(d)$ does not generalize directly to $S_+(r,d) \simeq \mathbb{R}_*^{d\times r}/O(r)$ because $\mathbb{R}_*^{d\times r}$ is not a group. However, a generalization is possible by considering the polar matrix factorization

$$\mathbf{G} = \mathbf{U}\mathbf{R}, \quad \mathbf{U} \in \mathrm{St}(r,d),\ \mathbf{R} \in S_+(r).$$

It is obtained from the singular value decomposition of $\mathbf{G} = \mathbf{Z\Sigma V}^T$ as $\mathbf{U} = \mathbf{ZV}^T$ and $\mathbf{R} = \mathbf{V\Sigma V}^T$ (Golub and Van Loan, 1996). This gives a polar parameterization of $S_+(r,d)$

$$\mathbf{W} = \mathbf{UR}^2\mathbf{U}^T.$$

This development leads to the quotient representation

$$S_+(r,d) \simeq (\mathrm{St}(r,d) \times S_+(r))/\mathcal{O}(r), \tag{15}$$

based on the invariance of $\mathbf{W}$ to the transformation $(\mathbf{U},\mathbf{R}^2) \mapsto (\mathbf{UO},\mathbf{O}^T\mathbf{R}^2\mathbf{O})$, $\mathbf{O} \in \mathcal{O}(r)$. It thus describes the set of equivalence classes

$$[(\mathbf{U},\mathbf{R}^2)] = \{(\mathbf{UO},\mathbf{O}^T\mathbf{R}^2\mathbf{O}) \text{ s.t. } \mathbf{O} \in \mathcal{O}(r)\}.$$

The cost function is now given by

$$f(\mathbf{U},\mathbf{R}^2) = \ell(\hat{y},y) = \frac{1}{2}(\mathrm{Tr}(\mathbf{UR}^2\mathbf{U}^T\mathrm{Sym}(\mathbf{X})) - y)^2. \tag{16}$$

The Riemannian geometry of (15) has been recently studied by Bonnabel and Sepulchre (2009). A tangent vector $\xi_{[\mathbf{W}]} = (\xi_{\mathbf{U}},\xi_{\mathbf{R}^2})_{[\mathbf{U},\mathbf{R}^2]}$ at $[(\mathbf{U},\mathbf{R}^2)]$ is described by a horizontal tangent vector $\bar{\xi}_{\mathbf{W}} = (\bar{\xi}_{\mathbf{U}},\bar{\xi}_{\mathbf{R}^2})_{(\mathbf{U},\mathbf{R}^2)}$ at $(\mathbf{U},\mathbf{R}^2)$ by

$$\bar{\xi}_{\mathbf{U}} = \Pi_{\mathbf{U}}\mathbf{\Delta},\ \mathbf{\Delta} \in \mathbb{R}^{d\times r}, \qquad \bar{\xi}_{\mathbf{R}^2} = \mathbf{R}\mathrm{Sym}(\mathbf{\Psi})\mathbf{R},\ \mathbf{\Psi} \in \mathbb{R}^{r\times r}.$$

The metric

$$\begin{aligned} g_{[\mathbf{W}]}(\xi_{[\mathbf{W}]},\zeta_{[\mathbf{W}]}) &\triangleq \bar{g}_{\mathbf{W}}(\bar{\xi}_{\mathbf{W}},\bar{\zeta}_{\mathbf{W}}) \\ &= \frac{1}{\lambda}\,\bar{g}_{\mathbf{U}}(\bar{\xi}_{\mathbf{U}},\bar{\zeta}_{\mathbf{U}}) + \frac{1}{1-\lambda}\,\bar{g}_{\mathbf{R}^2}(\bar{\xi}_{\mathbf{R}^2},\bar{\zeta}_{\mathbf{R}^2}), \end{aligned} \tag{17}$$

where $\lambda \in (0,1)$, is induced by the metric of $\mathrm{St}(r,d)$ and the affine-invariant metric of $S_+(r)$,

$$\bar{g}_{\mathbf{U}}(\mathbf{\Delta}_1,\mathbf{\Delta}_2) = \mathrm{Tr}(\mathbf{\Delta}_1^T\mathbf{\Delta}_2), \qquad \bar{g}_{\mathbf{R}^2}(\mathbf{\Psi}_1,\mathbf{\Psi}_2) = \mathrm{Tr}(\mathbf{\Psi}_1\mathbf{R}^{-2}\mathbf{\Psi}_2\mathbf{R}^{-2}).$$

The proposed metric is invariant along the set of equivalence classes and thus induces a quotient structure on $S_+(r,d)$. Alternative metrics on $S_+(r)$ can be considered as long as the metric remains invariant along the set of equivalence classes. For instance, the log-Euclidean metric discussed in Section 5.3 would qualify as a valid alternative.

A retraction is provided by distinct retractions on $\mathbf{U}$ and $\mathbf{R}^2$,

$$R_{\mathbf{U}}(s\bar{\xi}_{\mathbf{U}}) = \mathrm{qf}(\mathbf{U} + s\bar{\xi}_{\mathbf{U}}) \tag{18}$$
$$R_{\mathbf{R}^2}(s\bar{\xi}_{\mathbf{R}^2}) = \mathbf{R}\exp(s\mathbf{R}^{-1}\bar{\xi}_{\mathbf{R}^2}\mathbf{R}^{-1})\mathbf{R}. \tag{19}$$

One should observe that this retraction is not the exponential mapping of $S_+(r,d)$. This illustrates the interest of considering more general retractions than the exponential mapping. Indeed, as discussed in the paper of Bonnabel and Sepulchre (2009), the geodesics (and therefore the exponential

Batch regression

```
Input: ${(\mathbf{X}_i, y_i)}_{i=1}^n$
Require: $\mathbf{G}_0$ or $(\mathbf{U}_0, \mathbf{R}_0)$, $\lambda$
 1: $t = 0$
 2: repeat
 3:
 4:
 5:
 6:
 7:     Compute Armijo step $s_A$ from (22)
 8:     Perform update (10) or (21) using $s_A$
 9:
10:
11:     $t = t + 1$
12: until stopping criterion (24) is satisfied
13: return $\mathbf{G}_t$
```

Online regression

```
Input: ${(\mathbf{X}_t, y_t)}_{t \geq 1}$
Require: $\mathbf{G}_0$ or $(\mathbf{U}_0, \mathbf{R}_0)$, $\lambda$, $p$, $s$, $t_0$, $T$
 1: $t = 0, \texttt{count} = p$
 2: while $t \leq T$ do
 3:     if $\texttt{count} > 0$ then
 4:         Accumulate gradient
 5:         $\texttt{count} = \texttt{count} - 1$
 6:     else
 7:         Compute step size $s_t$ according to (23)
 8:         Perform update (9) or (20) using $s_t$
 9:         $\texttt{count} = p$
10:     end if
11:     $t = t + 1$
12: end while
13: return $\mathbf{G}_T$
```

Figure 2: Pseudo-codes for the proposed batch and online algorithms.

mapping) do not appear to have a closed form in the considered geometry. Combining the gradient of (16) with the retractions (18) and (19) gives

$$
\begin{aligned}
\mathbf{U}_{t+1} &= \mathrm{qf}\left(\mathbf{U}_t - 2\lambda s_t(\hat{y}_t - y_t)(\mathbf{I} - \mathbf{U}_t\mathbf{U}_t^T)\mathrm{Sym}(\mathbf{X}_t)\mathbf{U}_t\mathbf{R}_t^2\right),\\
\mathbf{R}_{t+1}^2 &= \mathbf{R}_t \exp\left(-(1-\lambda)s_t(\hat{y}_t - y_t)\mathbf{R}_t\mathbf{U}_t^T\mathrm{Sym}(\mathbf{X}_t)\mathbf{U}_t\mathbf{R}_t\right)\mathbf{R}_t.
\end{aligned}
$$

A factorization $\mathbf{R}_{t+1}\mathbf{R}_{t+1}^T$ of $\mathbf{R}_{t+1}^2$ is obtained thanks to the property of matrix exponential, $\exp(\mathbf{A})^{\frac{1}{2}} = \exp(\frac{1}{2}\mathbf{A})$. Updating $\mathbf{R}_{t+1}$ instead of $\mathbf{R}_{t+1}^2$ is thus more efficient from a computational point of view, since it avoids the computation of a square root a each iteration. This yields the online gradient descent algorithm

$$
\begin{aligned}
\mathbf{U}_{t+1} &= \mathrm{qf}\left(\mathbf{U}_t - 2\lambda s_t(\hat{y}_t - y_t)(\mathbf{I} - \mathbf{U}_t\mathbf{U}_t^T)\mathrm{Sym}(\mathbf{X}_t)\mathbf{U}_t\mathbf{R}_t^2\right),\\
\mathbf{R}_{t+1} &= \mathbf{R}_t \exp\left(-\frac{1}{2}(1-\lambda)s_t(\hat{y}_t - y_t)\mathbf{R}_t\mathbf{U}_t^T\mathrm{Sym}(\mathbf{X}_t)\mathbf{U}_t\mathbf{R}_t\right),
\end{aligned}
\tag{20}
$$

and the batch gradient descent algorithm

$$
\begin{aligned}
\mathbf{U}_{t+1} &= \mathrm{qf}\left(\mathbf{U}_t - 2\lambda s_t \frac{1}{n}\sum_{i=1}^n(\hat{y}_i - y_i)(\mathbf{I} - \mathbf{U}_t\mathbf{U}_t^T)\mathrm{Sym}(\mathbf{X}_i)\mathbf{U}_t\mathbf{R}_t^2\right),\\
\mathbf{R}_{t+1} &= \mathbf{R}_t \exp\left(-\frac{1}{2}(1-\lambda)s_t \frac{1}{n}\sum_{i=1}^n(\hat{y}_i - y_i)\mathbf{R}_t\mathbf{U}_t^T\mathrm{Sym}(\mathbf{X}_i)\mathbf{U}_t\mathbf{R}_t\right).
\end{aligned}
\tag{21}
$$

7. Algorithms

This section documents implementation details of the proposed algorithms. Generic pseudo-codes are provided in Figure 2 and Table 1 summarizes computational complexities.

Data type	Input space	Batch flat (10)	Batch polar (21)	Online flat (9)	Online polar (20)
$\mathbf{X}$	$\mathbb{R}^{d\times d}$	$O(d^2rn)$	$O(d^2r^2n)$	$O(d^2rp)$	$O(d^2r^2p)$
$\mathbf{x}\mathbf{x}^T$	$\mathbb{R}^d$	$O(drn)$	$O(dr^2n)$	$O(drp)$	$O(dr^2p)$

Table 1: Computational costs of the proposed algorithms.

7.1 From Subspace Learning to Distance Learning

The update expressions (21) and (20) show that λ, the tuning parameter of the Riemannian metric (17), acts as a weighting factor on the search direction. A proper tuning of this parameter allows us to place more emphasis either on the learning of the subspace $\mathbf{U}$ or on the distance in that subspace $\mathbf{R}^2$. In the case $\lambda = 1$, the algorithm only performs subspace learning. Conversely, in the case $\lambda = 0$, the algorithm learns a distance for a fixed range space (see Section 8.1). Intermediate values of λ continuously interpolate between the subspace learning problem and the distance learning problem at fixed range space.

A proper tuning of λ is of interest when a good estimate of the subspace is available (for instance a subspace given by a proper dimension reduction technique) or when too few observations are available to jointly estimate the subspace and the distance within that subspace. In the latter case, one has the choice to favor either subspace or distance learning.

Experimental results of Section 10 recommend the value $\lambda = 0.5$ as the default setting.

7.2 Invariance Properties

A nice property of the proposed algorithms is their invariance with respect to rotations $\mathbf{W} \mapsto \mathbf{O}^T\mathbf{W}\mathbf{O}$, $\forall \mathbf{O} \in O(d)$. This invariance comes from the fact that the chosen metrics are invariant to rotations. A practical consequence is that a rotation of the input matrix $\mathbf{X} \mapsto \mathbf{O}\mathbf{X}\mathbf{O}^T$ (for instance a whitening transformation of the vectors $\mathbf{x} \mapsto \mathbf{O}\mathbf{x}$ if $\mathbf{X} = \mathbf{x}\mathbf{x}^T$) will not affect the behavior of the algorithms.

Besides being invariant to rotations, algorithms (20) and (21) are invariant with respect to scalings $\mathbf{W} \mapsto \mu\mathbf{W}$ with $\mu > 0$. Consequently, a scaling of the input data $(\mathbf{X}, y) \mapsto (\mu\mathbf{X}, \mu y)$, such as a change of units, will not affect the behavior of these algorithms.

7.3 Mini-Batch Extension of Online Algorithms

We consider a mini-batch extension of stochastic gradient algorithms. It consists in performing each gradient step with respect to $p \geq 1$ examples at a time instead of a single one. This is a classical speedup and stabilization heuristic for stochastic gradient algorithms. In the particular case $p = 1$, one recovers plain stochastic gradient descent. Given p samples $(\mathbf{X}_{t,1}, y_{t,1}), \ldots, (\mathbf{X}_{t,p}, y_{t,p})$, received at time t, the abstract update (2) becomes

$$\mathbf{W}_{t+1} = R_{\mathbf{W}_t}\left(-s_t \, \frac{1}{p}\sum_{i=1}^{p} \operatorname{grad}\ell(\hat{y}_{t,i}, y_{t,i})\right).$$

7.4 Strategies for Choosing the Step Size

We here present strategies for choosing the step size in both the batch and online cases.

7.4.1 BATCH ALGORITHMS

For batch algorithms, classical backtracking methods exist (see Nocedal and Wright, 2006). In this paper, we use the Armijo step s_A defined at each iteration by the condition

$$f(R_{\mathbf{W}_t}(-s_A \operatorname{grad} f(\mathbf{W}_t))) \leq f(\mathbf{W}_t) + c\|\operatorname{grad} f(\mathbf{W}_t)\|^2_{\mathbf{W}_t}, \tag{22}$$

where $\mathbf{W}_t \in S_+(r,d)$ is the current iterate, $c \in (0,1)$, f is the empirical cost (1) and $R_\mathbf{W}$ is the chosen retraction. In this paper, we choose the particular value $c = 0.5$ and repetitively divide by 2 a specified maximum step size s_{max} until condition (22) is satisfied for the considered iteration. In order to reduce the dependence on s_{max} in a particular problem, it is chosen inversely proportional to the norm of the gradient at each iteration,

$$s_{max} = \frac{s_0}{\|\operatorname{grad} f(\mathbf{W}_t)\|_{\mathbf{W}_t}}.$$

A typical value of $s_0 = 100$ showed satisfactory results for all the considered problems.

7.4.2 ONLINE ALGORITHMS

For online algorithms, the choice of the step size is more involved. In this paper, the step size schedule s_t is chosen as

$$s_t = \frac{s}{\hat{\mu}_{grad}} \times \frac{nt_0}{nt_0 + t}, \tag{23}$$

where $s > 0$, n is the number of considered learning samples, $\hat{\mu}_{grad}$ is an estimate of the average gradient norm $\|\operatorname{grad} f(\mathbf{W}_0)\|_{\mathbf{W}_0}$, and $t_0 > 0$ controls the annealing rate of s_t. During a pre-training phase of our online algorithms, we select a small subset of learning samples and try the values 2^k with $k = -3,\dots,3$ for both s and t_0. The values of s and t_0 that provide the best decay of the cost function are selected to process the complete set of learning samples.

7.5 Stopping Criterion

Batch algorithms are stopped when the value or the relative change of the empirical cost f is small enough, or when the relative change in the parameter variation is small enough,

$$f(\mathbf{W}_{t+1}) \leq \varepsilon_{tol}, \quad \text{or} \quad \frac{f(\mathbf{W}_{t+1}) - f(\mathbf{W}_t)}{f(\mathbf{W}_t)} \leq \varepsilon_{tol}, \quad \text{or} \quad \frac{\|\mathbf{G}_{t+1} - \mathbf{G}_t\|_F}{\|\mathbf{G}_t\|_F} \leq \varepsilon_{tol}. \tag{24}$$

We found $\varepsilon_{tol} = 10^{-5}$ to be a good trade-off between accuracy and convergence time.

Online algorithms are run for a fixed number of epochs (number of passes through the set of learning samples). Typically, a few epochs are sufficient to attain satisfactory results.

7.6 Convergence

Gradient descent algorithms on matrix manifolds share the well-characterized convergence properties of their analog in $\mathbb{R}^d$. Batch algorithms converge linearly to a local minimum of the empirical cost that depends on the initial condition. Online algorithms converge asymptotically to a local minimum of the expected loss. They intrinsically have a much slower convergence rate than batch algorithms, but they generally decrease faster the expected loss in the large-scale regime (Bottou and Bousquet, 2007). The main idea is that, given a training set of samples, an inaccurate solution may indeed have the same or a lower expected cost than a well-optimized one.

When learning a matrix $\mathbf{W} \in S_+(d)$, the problem is convex and the proposed algorithms converge toward a global minimum of the cost function, regardless of the initial condition. When learning a low-rank matrix $\mathbf{W} \in S_+(r,d)$, with $r < d$, the proposed algorithms converge to a local minimum of the cost function. This is not the case for heuristic methods proposed in the literature, which first reduce the dimensionality of the data before fitting a full-rank model on the reduced data (Davis and Dhillon, 2008; Weinberger and Saul, 2009).

For batch algorithms, the local convergence results follow from the convergence theory of line-search algorithms on Riemannian manifolds (see, for example, Absil et al., 2008).

For online algorithms, one can prove that the algorithm based on the flat geometry enjoys almost sure asymptotic convergence to a local minimum of the expected cost. In that case, the parameter $\mathbf{G}$ belongs to an Euclidean space and the convergence results presented by Bottou (1998) apply (see Appendix A for the main ideas of the proof).

In contrast, when the polar parameterization is used, the convergence results presented by Bottou (1998) do not apply directly because of the quotient nature of the search space. Because the extension would require technical arguments beyond the scope of the present paper, we refrain from stating a formal convergence result for the online algorithm based on the polar geometry, even though the result is quite plausible.

Due to the nonconvex nature of the considered rank-constrained problems, the convergence results are only local and little can be presently said about the global convergence of the algorithms. A global analysis of the critical points of the cost functions studied in the present paper is nevertheless not hopeless and could be facilitated by the considered low-rank parameterizations. For instance, global convergence properties have been established for PCA algorithms from an explicit analysis of the critical points (Chen et al., 1998). Also, recent results suggest good global convergence properties for closely related rank minimization problems (Recht et al., 2010). Experimental results suggest the same conclusions for the algorithms considered in this paper, which means that further research on global convergence results is certainly deserved.

8. Discussion

This section presents connections with existing works and extensions of the regression model.

8.1 Closeness-Based Approaches

A standard derivation of learning algorithms is as follows (Kivinen and Warmuth, 1997). The (online) update at time t is viewed as an (approximate) solution of

$$\mathbf{W}_{t+1} = \underset{\mathbf{W} \in \mathcal{W}}{\arg\min}\; D(\mathbf{W}, \mathbf{W}_t) \;+\; s_t\, \ell(\hat{y}, y_t), \tag{25}$$

where D is a well-chosen measure of closeness between elements of $\mathcal{W}$ and s_t is a trade-off parameter that controls the balance between the conservative term $D(\mathbf{W}, \mathbf{W}_t)$ and the innovation (or data fitting) term $\ell(\hat{y}, y_t)$. One solves (25) by solving the algebraic equation

$$\operatorname{grad} D(\mathbf{W}, \mathbf{W}_t) = -s_t \operatorname{grad} \ell(\hat{y}_{t+1}, y_t), \tag{26}$$

which is a first-order (necessary) optimality condition. If the search space $\mathcal{W}$ is a Riemannian manifold and if the closeness measure $D(\mathbf{W},\mathbf{W}_t)$ is the Riemannian distance, the solution of (26) is

$$\mathbf{W}_{t+1} = \mathrm{Exp}_{\mathbf{W}_t}(-s_t \text{ grad } \ell(\hat{y}_{t+1}, y_t)).$$

Because $\hat{y}_{t+1}$ must be evaluated in $\mathbf{W}_{t+1}$, this update equation is implicit. However, $\hat{y}_{t+1}$ is generally replaced by $\hat{y}_t$ (which is equal to $\hat{y}_{t+1}$ up to first order terms in s_t), which gives the update (2) where the exponential mapping is chosen as a retraction.

Bregman divergences have been popular closeness measures for $D(\mathbf{W},\mathbf{W}_t)$ because they render the optimization of (25) convex. Bregman divergences on the cone of positive definite matrices include the von Neumann divergence

$$D_{vN}(\mathbf{W},\mathbf{W}_t) = \mathrm{Tr}(\mathbf{W}\log\mathbf{W} - \mathbf{W}\log\mathbf{W}_t - \mathbf{W} + \mathbf{W}_t),$$

and the LogDet divergence

$$D_{ld}(\mathbf{W},\mathbf{W}_t) = \mathrm{Tr}(\mathbf{W}\mathbf{W}_t^{-1}) - \log\det(\mathbf{W}\mathbf{W}_t^{-1}) - d.$$

We have shown in Section 5 that the resulting updates can be interpreted as line-search updates for the log-Euclidean metric and the affine-invariant metric of $S_+(d)$ and for specific choices of the retraction mapping.

Likewise, the algorithm (9) can be recast in the framework (25) by considering the closeness

$$D_{flat}(\mathbf{W},\mathbf{W}_t) = \|\mathbf{G} - \mathbf{G}_t\|_F^2,$$

where $\mathbf{W} = \mathbf{G}\mathbf{G}^T$ and $\mathbf{W}_t = \mathbf{G}_t\mathbf{G}_t^T$. Algorithm (20) can be recast in the framework (25) by considering the closeness

$$D_{pol}(\mathbf{W},\mathbf{W}_t) = \lambda \sum_{i=1}^{r} \theta_i^2 \; + \; (1-\lambda)\,\|\log\mathbf{R}_t^{-1}\mathbf{R}^2\mathbf{R}_t^{-1}\|_F^2.$$

where the θ_i's are the principal angles between the subspaces spanned by $\mathbf{W}$ and $\mathbf{W}_t$ (Golub and Van Loan, 1996), and the second term is the affine-invariant distance of $S_+(d)$ between matrices $\mathbf{R}^2$ and $\mathbf{R}_t^2$ involved in the polar representation of $\mathbf{W}$ and $\mathbf{W}_t$.

Obviously, these closeness measures are no longer convex due to the rank constraint. However they reduce to the popular divergences in the full-rank case, up to second order terms. In particular, when $\lambda = 1$, the subspace is fixed and one recovers the setup of learning low-rank matrices of a fixed range space (Kulis et al., 2009). Thus, the algorithms introduced in the present paper can be viewed as generalizations of the ones presented in the paper of Kulis et al. (2009), where the issue of adapting the range space is presented as an open research question. Each of the proposed algorithms provides an efficient workaround for this problem at the expense of the (potential) introduction of local minima.

8.2 Handling Inequalities

Inequalities $\hat{y} \leq y$ or $\hat{y} \geq y$ can be considered by treating them as equalities when they are not satisfied. This is equivalent to the minimization of the continuously differentiable cost function

$$f(\mathbf{W}) = \ell(\hat{y}, y) = \frac{1}{2}\max(0, \rho(\hat{y} - y))^2,$$

where $\rho = +1$ if $\hat{y} \leq y$ is required and $\rho = -1$ if $\hat{y} \geq y$ is required.

8.3 Kernelizing the Regression Model

In this paper, we have not considered the kernelized model

$$\hat{y} = \mathrm{Tr}(\mathbf{W}\phi(\mathbf{x})\phi(\mathbf{x})^T),$$

whose predictions can be extended to new input data $\phi(\mathbf{x})$ in the feature space $\mathcal{F}$ induced by the nonlinear mapping $\phi : \mathbf{x} \in \mathcal{X} \mapsto \phi(\mathbf{x}) \in \mathcal{F}$. This is potentially a useful extension of the regression model that could be investigated in the light of recent theoretical results in this area (for example Chatpatanasiri et al., 2010; Jain et al., 2010).

8.4 Connection with Multidimensional Scaling Algorithms

Given a set of m dissimilarity measures $\mathcal{D} = \{\delta_{ij}\}^m$ between n data objects, multidimensional scaling algorithms search for a r-dimensional embedding of the data objects into an Euclidean space representation $\mathbf{G} \in \mathbb{R}^{n\times r}$ (Cox and Cox, 2001; Borg and Groenen, 2005). Each row $\mathbf{g}$ of $\mathbf{G}$ is the coordinates of a data object in a Euclidean space of dimension r.

Multidimensional scaling algorithms based on gradient descent are equivalent to algorithms (9) and (10) when $\mathbf{X} = (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T$, where $\mathbf{e}_i$ is the i-th unit vector (see Section 9.1), and when the multidimensional scaling reduction criterion is the SSTRESS

$$SSTRESS(\mathbf{G}) = \sum_{(i,j)\in\mathcal{D}} (\|\mathbf{g}_i - \mathbf{g}_j\|_2^2 - \delta_{ij})^2.$$

Vectors $\mathbf{g}_i$ and $\mathbf{g}_j$ are the i-th and j-th rows of matrix $\mathbf{G}$. Gradient descent is a popular technique in the context of multidimensional scaling algorithms. A stochastic gradient descent approach for minimizing the SSTRESS has also been proposed by Matsuda and Yamaguchi (2001). A potential area of future work is the application of the proposed online algorithm (9) for adapting a batch solution to slight modifications of the dissimilarities over time. This approach has a much smaller computational cost than recomputing the offline solution at every time step. It further allows to keep the coordinate representation coherent over time since the solution do not brutally jumps from a local minimum to another.

9. Applications

The choice of an appropriate distance measure is a central issue for many distance-based classification and clustering algorithms such as nearest neighbor classifiers, support vector machines or k-means. Because this choice is highly problem-dependent, numerous methods have been proposed to learn a distance function directly from data. In this section, we present two important distance learning applications that are compatible with the considered regression model and review some relevant literature on the subject.

9.1 Kernel Learning

In kernel-based methods (Shawe-Taylor and Cristianini, 2004), the data samples $\mathbf{x}_1,...,\mathbf{x}_n$ are first transformed by a nonlinear mapping $\phi : \mathbf{x} \in \mathcal{X} \mapsto \phi(\mathbf{x}) \in \mathcal{F}$, where $\mathcal{F}$ is a new feature space that is expected to facilitate pattern detection into the data.

The kernel function is then defined as the dot product between any two samples in $\mathcal{F}$,

$$\kappa(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j).$$

In practice, the kernel function is represented by a positive semidefinite matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ whose entries are defined as $\mathbf{K}_{ij} = \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j)$. This inner product information is used solely to compute the relevant quantities needed by the algorithms based on the kernel. For instance, a distance is implicitly defined by any kernel function as the Euclidean distance between the samples in the new feature space

$$d_\phi(\mathbf{x}_i,\mathbf{x}_j) = \|\phi(\mathbf{x}_i)-\phi(\mathbf{x}_j)\|^2 = \kappa(\mathbf{x}_i,\mathbf{x}_i)+\kappa(\mathbf{x}_j,\mathbf{x}_j)-2\kappa(\mathbf{x}_i,\mathbf{x}_j),$$

which can be evaluated using only the elements of the kernel matrix by the formula

$$d_\phi(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{K}_{ii}+\mathbf{K}_{jj}-2\mathbf{K}_{ij} = \operatorname{Tr}\left(\mathbf{K}(\mathbf{e}_i-\mathbf{e}_j)(\mathbf{e}_i-\mathbf{e}_j)^T\right),$$

which fits into the considered regression model.

Learning a kernel consists in computing the kernel (or Gram) matrix from scratch or improving a existing kernel matrix based on side-information (in a semi-supervised setting for instance). Data samples and class labels are generally exploited by means of equality or inequality constraints involving pairwise distances or inner products.

Most of the numerous kernel learning algorithms that have been proposed work in the so-called transductive setting, that is, it is not possible to generalize the learned kernel function to new data samples (Kwok and Tsang, 2003; Lanckriet et al., 2004; Tsuda et al., 2005; Zhuang et al., 2009; Kulis et al., 2009). In that setting, the total number of considered samples is known in advance and determines the size of the learned matrix. Recently, algorithms have been proposed to learn a kernel function that can be extended to new points (Chatpatanasiri et al., 2010; Jain et al., 2010). In this paper, we only consider the kernel learning problem in the transductive setting.

When low-rank matrices are considered, kernel learning algorithms can be regarded as dimensionality reduction methods. Very popular unsupervised algorithms in that context are kernel principal component analysis (Schölkopf et al., 1998) and multidimensional scaling (Cox and Cox, 2001; Borg and Groenen, 2005). Other kernel learning techniques include the maximum variance unfolding algorithm (Weinberger et al., 2004) and its semi-supervised version (Song et al., 2007), and the kernel spectral regression framework (Cai et al., 2007) which encompasses many reduction criterion (for example, linear discriminant analysis (LDA), locality preserving projection (LPP), neighborhood preserving embedding (NPE)). See the survey of Yang (2006) for a more complete state-of-the-art in this area.

Since our algorithms are able to compute a low-rank kernel matrix from data, they can be used for unsupervised or semi-supervised dimensionality reduction, depending whether or not the class labels are exploited through the imposed constraints.

9.2 Mahalanobis Distance Learning

Mahalanobis distances generalize the usual Euclidean distance as it allows to transform the data with an arbitrary rotation and scaling before computing the distance. Let $\mathbf{x}_i,\mathbf{x}_j \in \mathbb{R}^d$ be two data samples, the (squared) Mahalanobis distance between these two samples is parameterized by a positive definite matrix $\mathbf{A}\in\mathbb{R}^{d\times d}$ and writes as

$$d_\mathbf{A}(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i-\mathbf{x}_j)^T\mathbf{A}\,(\mathbf{x}_i-\mathbf{x}_j). \tag{27}$$

In the particular case of $\mathbf{A}$ being equal to the identity matrix, the standard Euclidean distance is obtained. A frequently used matrix is $\mathbf{A}=\mathbf{\Sigma}^{-1}$, the inverse of the sample covariance matrix. For centered data features, computing this Mahalanobis distance is equivalent to perform a whitening of the data before computing the Euclidean distance.

For low-rank Mahalanobis matrices, computing the distance is equivalent to first perform a linear data reduction step before computing the Euclidean distance on the reduced data.[1] Learning a low-rank Mahalanobis matrix can thus be seen as learning a linear projector that is used for dimension reduction.

In contrast to kernel functions, Mahalanobis distances easily generalize to new data samples since the sole knowledge of $\mathbf{A}$ determines the distance function.

In recent years, Mahalanobis distance learning algorithms have been the subject of many contributions that cannot be all enumerated here. We review a few of them, most relevant for the present paper. The first proposed methods have been based on successive projections onto a set of large margin constraints (Xing et al., 2002; Shalev-Shwartz et al., 2004). The method proposed by Globerson and Roweis (2005) seeks a Mahalanobis matrix that maximizes the between classes distance while forcing to zero the within classes distance. A simpler objective is pursued by the algorithms that optimize the Mahalanobis distance for the specific k-nearest neighbor classifier (Goldberger et al., 2004; Torresani and Lee, 2006; Weinberger and Saul, 2009). Bregman projection based methods minimize a particular Bregman divergence under distance constraints. Both batch (Davis et al., 2007) and online (Jain et al., 2008) formulations have been proposed for learning full-rank matrices. Low-rank matrices have also been considered with Bregman divergences but only when the range space of the matrix is fixed in the first place (Davis and Dhillon, 2008; Kulis et al., 2009).

10. Experiments

Data Set	Samples	Features	Classes	Reference
GyrB	52	-	3	Tsuda et al. (2005)
Digits	300	16	3	Asuncion and Newman (2007)
Wine	178	13	13	Asuncion and Newman (2007)
Ionosphere	351	33	2	Asuncion and Newman (2007)
Balance Scale	625	4	3	Asuncion and Newman (2007)
Iris	150	4	3	Asuncion and Newman (2007)
Soybean	532	35	17	Asuncion and Newman (2007)
USPS	2,007	256	10	LeCun et al. (1989)
Isolet	7,797	617	26	Asuncion and Newman (2007)
Prostate	322	15,154	2	Petricoin et al. (2002)

Table 2: Considered data sets

In this section, we illustrate the potential of the proposed algorithms on several benchmark experiments. First, the proposed algorithms are evaluated on toy data. Then, they are compared to state-of-the-art kernel learning and Mahalanobis distance learning algorithms on real data sets. Overall, the experiments support that a joint estimation of a subspace and low-dimensional distance in that subspace is a major advantage of the proposed algorithms over methods that estimate the matrix for a subspace that is fixed beforehand.

1. In the low-rank case, one should rigorously refer to (27) as a pseudo-distance. Indeed, one has $d_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) = 0$ with $\mathbf{x}_i \neq \mathbf{x}_j$ whenever $(\mathbf{x}_i - \mathbf{x}_j)$ lies in the null space of $\mathbf{A}$.

Table 2 summarizes the different data sets that have been considered. As a normalization step, the data features are centered and rescaled to unit standard deviation.

The implementation of the proposed algorithms,[2] as well as the experiments of this paper are performed with Matlab. The implementations of algorithms MVU,[3] KSR,[4] LMNN,[5] and ITML,[6] have been rendered publicly available by Weinberger et al. (2004), Cai et al. (2007), Weinberger and Saul (2009) and Davis et al. (2007) respectively. Algorithms POLA (Shalev-Shwartz et al., 2004), LogDet-KL (Kulis et al., 2009) and LEGO (Jain et al., 2008) have been implemented on our own.

10.1 Toy Data

In this section, the proposed algorithms are evaluated on synthetic regression problems. The data vectors $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$ and the target matrix $\mathbf{W}^* \in S_+(r,d)$ are generated with entries drawn from a standard Gaussian distribution $\mathcal{N}(0,1)$. Observations follow

$$y_i = (\mathbf{x}_i^T \mathbf{W}^* \mathbf{x}_i)(1+\nu_i), \quad i = 1, ..., n, \tag{28}$$

where ν_i is drawn from $\mathcal{N}(0, 0.01)$. A multiplicative noise model is preferred over an additive one to easily control that observations remain nonnegative after the superposition of noise.

10.1.1 LEARNING THE SUBSPACE VS. FIXING THE SUBSPACE UP FRONT

As an illustrative example, we show the difference between two approaches for fitting the data to observations when a target model $\mathbf{W}^* \in S_+(3,3)$ is approximated with a parameter $\mathbf{W} \in S_+(2,3)$.

A naive approach to tackle that problem is to first project the data $\mathbf{x}_i \in \mathbb{R}^3$ on a subspace of reduced dimension and then to compute a full-rank model based on the projected data. Recent methods compute that subspace of reduced dimension using principal component analysis (Davis and Dhillon, 2008; Weinberger and Saul, 2009), that is, a subspace that captures a maximal amount of variance in the data. However, in general, there is no reason why the subspace spanned by the top principal components should coincide with the subspace that is defined by the target model. Therefore, a more appropriate approach consists in learning jointly the subspace and a distance in that subspace that best fits the data to observations within that subspace.

To compare the two approaches, we generate a set of learning samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{200}$, with $\mathbf{x}_i \in \mathbb{R}^3$ and y_i that follows (28). The target model is

$$\mathbf{W}^* = \tilde{\mathbf{U}} \mathbf{\Lambda} \tilde{\mathbf{U}}^T$$

where $\tilde{\mathbf{U}}$ is a random 3×3 orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix with two dominant values $\Lambda_{11}, \Lambda_{22} \gg \Lambda_{33} > 0$ (for this specific example, $\Lambda_{11} = 4, \Lambda_{22} = 3$ and $\Lambda_{33} = 0.01$). Observations y_i are thus nearly generated by a rank-2 model, such that $\mathbf{W}^*$ should be well approximated with a matrix $\mathbf{W} \in S_+(2,3)$ that minimizes the train error.

Results are presented in Figure 3. The top plot shows that the learned subspace (which identifies with the target subspace) is indeed very different from the subspace spanned by the top two principal components. Moreover, the bottom plots clearly demonstrate that the fit is much better when the

2. The source code is available from `http://www.montefiore.ulg.ac.be/~meyer`.
3. MVU is available from `http://www.cse.wustl.edu/~kilian/Downloads/MVU.html`.
4. KSR is available from `http://www.cs.uiuc.edu/homes/dengcai2/SR/`.
5. LMNN is available from `http://www.cse.wustl.edu/~kilian/Downloads/LMNN.html`.
6. ITML is available from `http://www.cs.utexas.edu/users/pjain/itml/`.

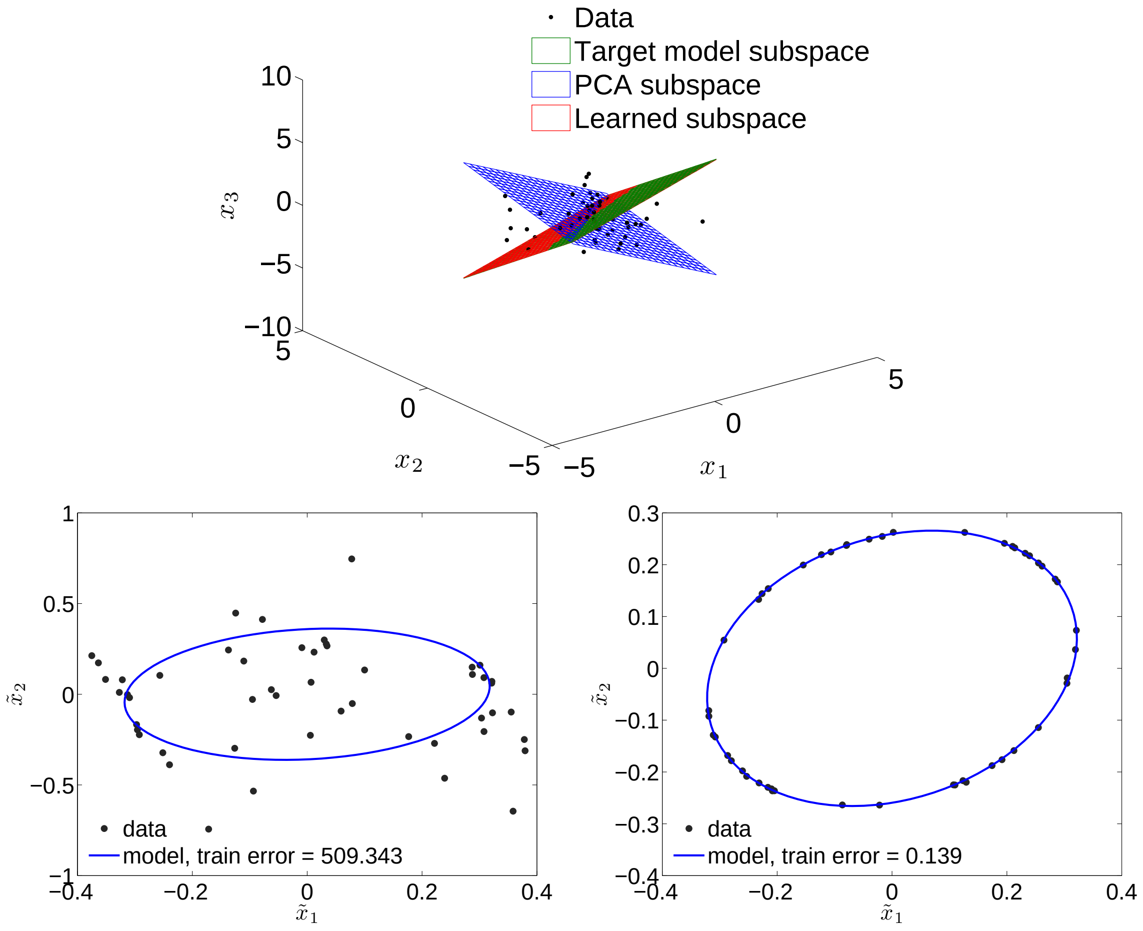


Figure 3: Learning vs fixing the subspace. **Top**: the learned subspace is very different from the subspace computed from a classical heuristic. **Bottom left**: fit after projection of the data onto a subspace fixed up front. **Bottom right**: fit obtained with a join estimation of the subspace and a distance within that subspace.

subspace and the distance in that subspace are learned jointly. The difference is also significant in terms of the train error. This simple example shows that heuristic methods that fix the range space in the first place may converge to a solution that is very different from a minimum of the desired cost function. For visualization purpose, the two dimensional model is represented by the ellipse

$$\mathcal{E} = \{\tilde{\mathbf{x}}_i \in \mathbb{R}^2 : \tilde{\mathbf{x}}_i^T \mathbf{R}^2 \tilde{\mathbf{x}}_i = 1\}, \qquad \text{where} \quad \tilde{\mathbf{x}}_i = \frac{\mathbf{U}^T \mathbf{x}_i}{\sqrt{y_i}},$$

and $(\mathbf{U}, \mathbf{R}^2)$ are computed with algorithm (21), either in the setting $\lambda = 0$ that fixes the subspace to the PCA subspace (left) or in the setting $\lambda = 0.5$ that simultaneously learned $\mathbf{U}$ and $\mathbf{B}$ (right). A perfect fit is obtained when all $\tilde{\mathbf{x}}_i$ are located on $\mathcal{E}$, which is the locus of points where $\hat{y}_i = y_i$.

10.1.2 INFLUENCE OF λ ON THE ALGORITHM BASED ON THE POLAR GEOMETRY

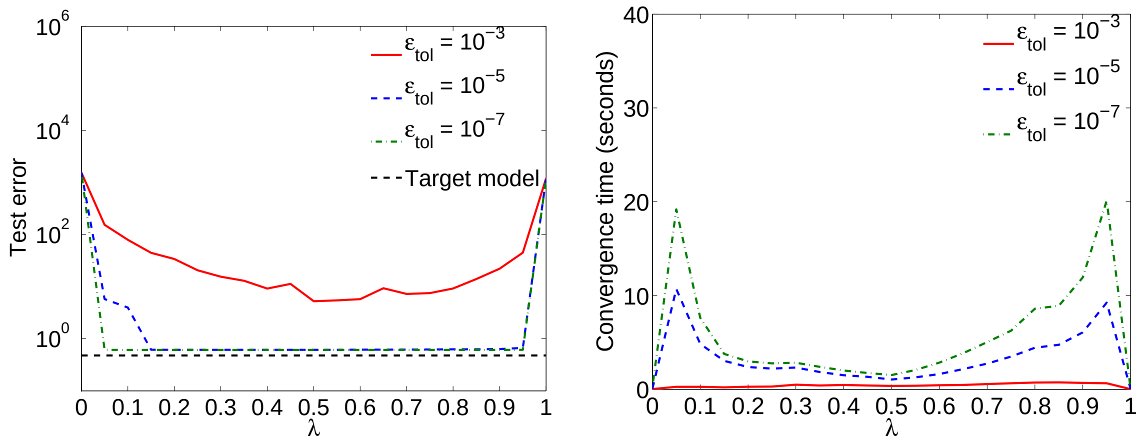


Figure 4: Influence of λ.

In theory, the parameter λ should not influence the algorithm since it has no effect on the first-order optimality conditions except for its two extreme values $\lambda = 0$ and $\lambda = 1$. In practice however, a sensitivity to this parameter is observed due to the finite tolerance of the stopping criterion: the looser the tolerance, the more sensitive to λ.

To investigate the sensitivity to λ, we try to recover a target parameter $\mathbf{W}^* \in S_+(5,10)$ using pairs $(\mathbf{x}_i, y_i)$ generated according to (28). We generate 10 random regression problems with 1000 samples partitioned into 500 learning samples and 500 test samples. We compute the mean test error and the mean convergence time as a function of λ for different values of ε_{tol}. The results are presented in Figure 4. As ε_{tol} decrease, the test error becomes insensitive to λ, but an influence is observed on the convergence time of the algorithm.

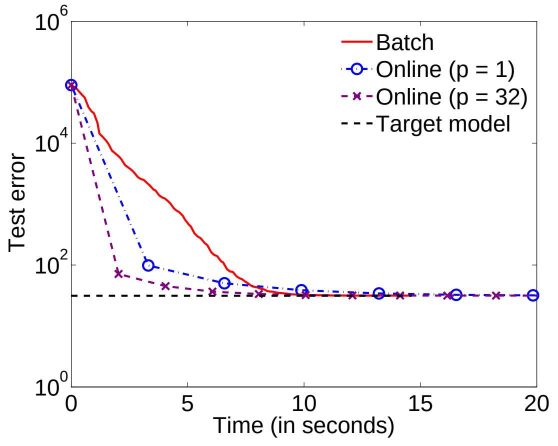
In view of these results, we recommend the value 0.5 as the default setting for λ. Unless specified otherwise, we therefore use this particular value for all experiments in this paper.

10.1.3 ONLINE VS. BATCH

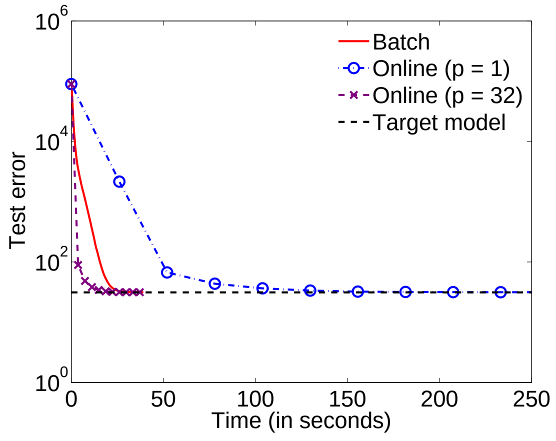
This experiment shows that when a large amount of sample is available (80,000 training samples and 20,000 test samples for learning a parameter $\mathbf{W}^*$ in $S_+(10,50)$), online algorithms minimize the test error more rapidly than batch ones. It further shows that the mini-batch extension allows to improve significantly the performance compared to the plain stochastic gradient descent setting ($p = 1$). We observe that the mini-batch size $p = 32$ generally gives good results. Figure 5 report the test error as a function of the learning time, that is, the time after each iteration for batch algorithm and the time after each epoch for online algorithms. For the algorithm based on the polar geometry, the mini-batch extension is strongly recommended to amortize the larger cost of each update.

10.2 Kernel Learning

In this section, the proposed algorithms are applied to the problem of learning a kernel matrix from pairwise distance constraints between data samples. As mentioned earlier, we only consider



(a) Flat geometry



(b) Polar geometry

Figure 5: Online vs Batch. For a large number of samples, online algorithms reduce the test error much more rapidly than batch ones. Using the mini-batch extension generally improve significantly the performance.

this problem in the transductive setting, that is, all samples $\mathbf{x}_1, \ldots \mathbf{x}_n$ are available up front and the learned kernel do not generalize to new samples.

10.2.1 EXPERIMENTAL SETUP

After transformation of the data with the kernel map $\mathbf{x} \mapsto \phi(\mathbf{x})$, the purpose is to compute a fixed-rank kernel matrix based on a limited amount of pairwise distances in the kernel feature space and on some information about class labels.

Distance constraints are generated as $\hat{y}_{ij} \leq y_{ij}(1-\alpha)$ for identically labeled samples and $\hat{y}_{ij} \geq y_{ij}(1+\alpha)$ for differentially labeled samples, where $\alpha \geq 0$ is a scaling factor, $y_{ij} = \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2$ and $\hat{y}_{ij} = \mathrm{Tr}(\mathbf{W}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T) = (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{W}(\mathbf{e}_i - \mathbf{e}_j)$.

We investigate both the influence of the amount of side-information provided, the influence of the approximation rank and the computational time required by the algorithms.

To quantify the performance of the learned kernel matrix, we perform either a classification or a clustering of the samples based on the learned kernel. For classification, we compute the test set accuracy of a k-nearest neighbor classifier ($k = 5$) using a two-fold cross-validation protocol (results are averaged over 10 random splits). For clustering, we use the K-means algorithm with the number of clusters equal to the number of classes in the problem. To overcome K-means local minima, 10 runs are performed in order to select the result that has lead to the smaller value of the K-means objective. The quality of the clustering is measured by the normalized mutual information (NMI) shared between the random variables of cluster indicators C and target labels T (Strehl et al., 2000),

$$NMI = \frac{2\, I(C;T)}{(H(C) + H(T))},$$

where $I(X_1;X_2) = H(X_1) - H(X_1|X_2)$ is the mutual information between the random variables X_1 and X_2, $H(X_1)$ is the Shannon entropy of X_1, and $H(X_1|X_2)$ is the conditional entropy of X_1 given X_2. This score ranges from 0 to 1, the larger the score, the better the clustering quality.

10.2.2 COMPARED METHODS

We compare the following methods:

1. Batch algorithms (10) and (21), adapted to handle inequalities (see Section 8.2),

2. The kernel learning algorithm LogDet-KL (Kulis et al., 2009) which learn kernel matrices of fixed range space for a given set of distance constraints.

3. The kernel spectral regression (KSR) algorithm of Cai et al. (2007) using a similarity matrix $\mathbf{N}$ constructed as follows. Let $\mathbf{N}$ be the adjacency matrix of a 5-NN graph based on the initial kernel. We modify $\mathbf{N}$ according to the set of available constraints: $\mathbf{N}_{ij} = 1$ if samples $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class (must-link constraint), $\mathbf{N}_{ij} = 0$ if samples $\mathbf{x}_i$ and $\mathbf{x}_j$ do not belong to the same class (cannot-link constraint).

4. The Maximum Variance Unfolding (MVU) algorithm (Weinberger et al., 2004),

5. The Kernel PCA algorithm (Schölkopf et al., 1998).

The last two algorithms are unsupervised techniques that are provided as baselines.

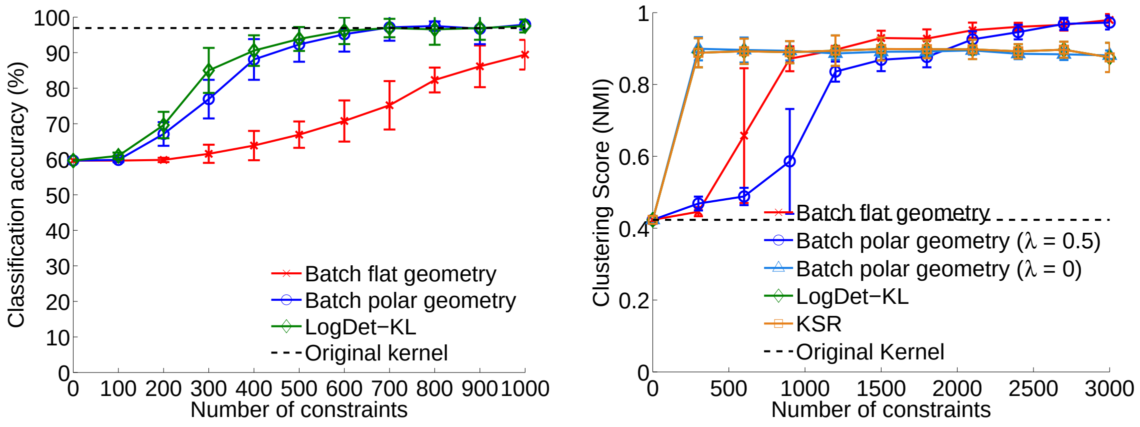


Figure 6: **Left**: full-rank kernel learning on the Gyrb data set. The algorithm based on the polar geometry competes with LogDet-KL. **Right**: low-rank kernel learning on the Digits data set. The proposed algorithms outperform the compared methods as soon as a sufficiently large number of constraints is provided.

10.2.3 RESULTS

The first experiment is reproduced from Tsuda et al. (2005) and Kulis et al. (2009). The goal is to reconstruct the GyrB kernel matrix based on distance constraints only. This matrix contains information about the proteins of three bacteria species. The distance constraints are randomly generated from the original kernel matrix with $\alpha = 0$. We compare the proposed batch methods with the LogDet-KL algorithm, the only competing algorithm that also learns directly from distance constraints. This algorithm is the best performer reported by Kulis et al. (2009) for this experiment. All algorithms start from the identity matrix that do not encode any domain information. Figure 6 (left) reports the *k*-NN classification accuracy as a function of the number of distance constraints provided. In this full-rank learning setting, the algorithm based on the polar geometry compete with the LogDet-KL algorithm. The convergence time of the algorithm based on the polar geometry is however much faster (0.15 seconds versus 58 seconds for LogDet-KL when learning 1000 constraints). The algorithm based on the flat geometry has inferior performance when too few constraints are provided. This is because in the kernel learning setting, updates of this algorithm only involve the rows and columns that correspond to the set of points for which constraints are provided. It may thus result in a partial update of the kernel matrix entries. This issue disappears as the number of provided constraints increases.

The second experiment is reproduced from Kulis et al. (2009). It aims at improving an existing low-rank kernel using limited information about class labels. A rank-16 kernel matrix is computed for clustering a database of 300 handwritten digits randomly sampled from the 3, 8 and 9 digits of the Digits data set (since we could not find out the specific samples that have been selected by Kulis et al. (2009), we made our own samples selection). The distance constraints are randomly sampled from a linear kernel on the input data $\mathbf{K} = \mathbf{X}\mathbf{X}^T$ and $\alpha = 0.25$. The results are presented in Figure 6 (right). The figure shows that KSR, LogDet-KL and the algorithm based on the polar geometry with $\lambda = 0$ perform similarly. These methods are however outperformed by the proposed algorithms (flat geometry and polar geometry with $\lambda = 0.5$) when the number of constraints is large enough. This experiment also enlightens the flexibility of the polar geometry, which allows us to fix the subspace in situations where too few constraints are available.

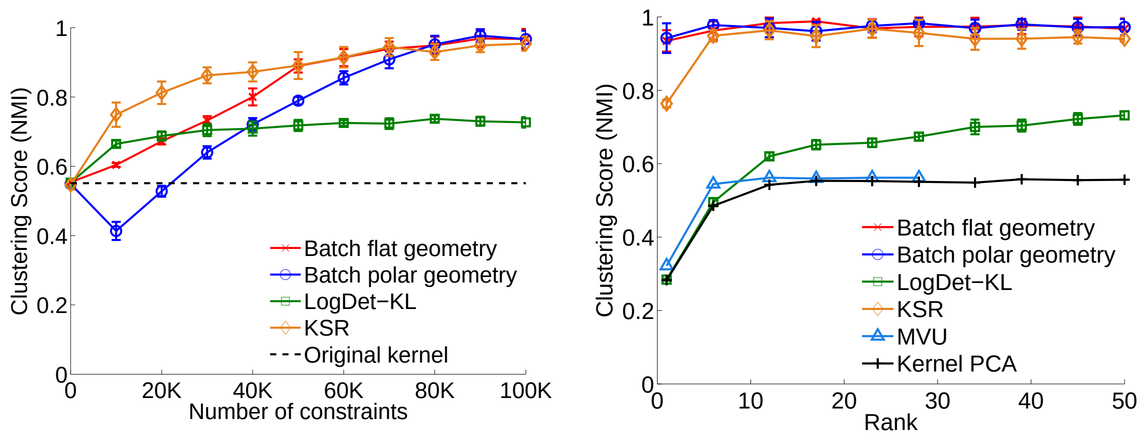


Figure 7: Clustering the USPS data set. **Left:** clustering score versus number of constraints. **Right:** clustering score versus approximation rank. When the number of provided constraints is large enough, the proposed algorithms perform as good as the KSR algorithm. It outperforms the LogDet-KL algorithm and baselines.

Finally, we tackle the kernel learning problem on a larger data set. We use the test set of the USPS data set,[7] which contains 2007 samples of handwritten zip code digits. The data are first transformed using the kernel map $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|_2^2)$ with $\gamma = 0.001$ and we further center the data in the kernel feature space. Pairwise distance constraints are randomly sampled from that kernel matrix with $\alpha = 0.5$. Except KSR that has its own initialization procedure, algorithms start from the kernel matrix provided by kernel PCA.

Figure 7 (left) shows the clustering performance as a function of the number of constraints provided when the approximation rank is fixed to $r = 25$. Figure 7 (right) reports the clustering performance as a function of the approximation rank when the number of constraints provided is fixed to $100K$. When the number of provided constraints is large enough, the proposed algorithms perform as good as KSR and outperform the LogDet-KL method that learn a kernel of fixed-range space. Average computational times for learning a rank-6 kernel from $100K$ constraints are 0.57 seconds for KSR, 3.25 seconds for the algorithm based on the flat geometry, 46.78 seconds for LogDet-KL and 47.30 seconds for the algorithm based on the polar geometry. In comparison, the SDP-based MVU algorithm takes 676.60 seconds to converge.

10.3 Mahalanobis Distance Learning

In this section, we tackle the problem of learning from data a Mahalanobis distance for supervised classification and compare our methods to state-of-the-art Mahalanobis metric learning algorithms.

10.3.1 Experimental Setup

For the considered problem, the purpose is to learn the parameter $\mathbf{W}$ of a Mahalanobis distance $d_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j)$, such that the distance satisfies as much as possible a given set of constraints. As in the paper of Davis et al. (2007), we generate the constraints from the learning set of samples as $d_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) \leq l$ for same-class pairs and $d_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) \geq u$ for different-class pairs. The scalars u and l estimate the 95-th and 5-th percentiles of the distribution of Mahalanobis distances parameterized by a chosen baseline $\mathbf{W}_0$. The performance of the learned distance is then quantified by the test error rate of a k-nearest neighbor classifier based on the learned distance. All experiments use the setting $k = 5$, breaking ties arbitrarily. Unless for the Isolet data set for which a specific train/test partition is provided, error rates are computed using two-fold cross validation. Results are averaged over 10 random partitions.

10.3.2 Compared Methods

We compare the following distance learning algorithms:

1. Batch algorithms (10) and (21),
2. ITML (Davis et al., 2007),
3. LMNN (Weinberger and Saul, 2009),
4. Online algorithms (9) and (20),
5. LEGO (Jain et al., 2008),

7. We use the ZIP code data from `http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/data.html`.

6. POLA (Shalev-Shwartz et al., 2004).

When some methods require the tuning of an hyper-parameter, this is performed by a two-fold cross-validation procedure. The slack parameter of ITML as well as the step size of POLA are selected in the range of values 10^k with $k = -3,\dots,3$. The step size of LEGO is selected in this same range of value for the UCI data sets, and in the range of value 10^k with $k = -10,\dots,-5$ for the larger data sets Isolet and Prostate.

10.3.3 RESULTS

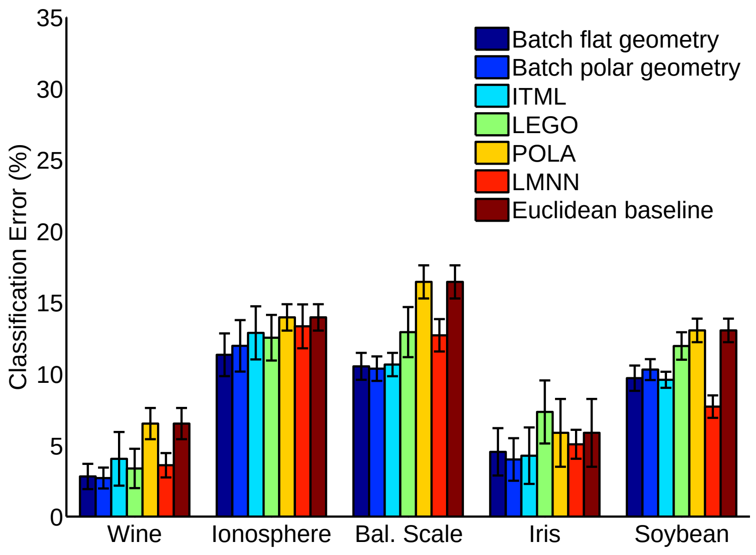
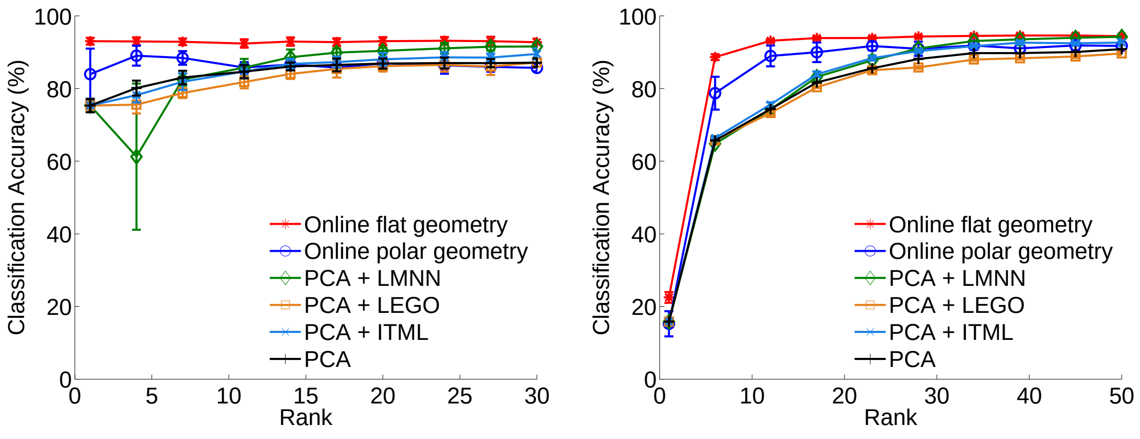


Figure 8: Full-rank distance learning on the UCI data sets. The proposed algorithms compete with state-of-the-art methods for learning a full-rank Mahalanobis distance.

Reproducing a classical benchmark experiment from Kulis et al. (2009), we demonstrate that the proposed batch algorithms compete with state-of-the-art full-rank Mahalanobis distance learning algorithms on several UCI data sets (Figure 8). We have not included the online versions of our algorithms in this comparison because we consider that the batch approaches are more relevant on such small data sets. Except POLA and LMNN which do not learn from provided pairwise constraints, all algorithms process $40c(c-1)$ constraints, where c is the number of classes in the data. We choose the Euclidean distance ($\mathbf{W}_0 = \mathbf{I}$) as the baseline distance for initializing the algorithms. Figure 8 reports the results. The two proposed algorithms compete favorably with the other full-rank distance learning techniques, achieving the minimal average error for 4 of the 5 considered data sets.

We finally evaluate the proposed algorithms on higher-dimensional data sets in the low-rank regime (Figure 9). The distance constraints are generated as in the full-rank case, but the initial baseline matrix is now computed as $\mathbf{W}_0 = \mathbf{G}_0\mathbf{G}_0^T$, where $\mathbf{G}_0$'s columns are the top principal directions of the data. For the Isolet data set, $100K$ constraints are generated, and $10K$ constraints are generated for the Prostate data set. For scalability reasons, algorithms LEGO, LMNN and ITML must proceed in two steps: the data are first projected onto the top principal directions and then



(a) Prostate (b) Isolet

Figure 9: Low-rank Mahalanobis distance learning. For low values of the rank, the proposed algorithms perform much better than the methods that project the data on the top principal directions and learn a full-rank distance on the projected data.

a full-rank distance is learned within the subspace spanned by these top principal directions. In contrast, our algorithms are initialized with the top principal direction, but they operate on the data in their original feature space. Overall, the proposed algorithms achieve much better performance than the methods that first reduce the data. This is particularly striking when the rank is very small compared to problem size. The performance gap reduces as the rank increases. However, for high-dimensional problems, one is usually interested in efficient low-rank approximations that gives satisfactory results.

11. Conclusion

In this paper, we propose gradient descent algorithms to learn a regression model parameterized by a fixed-rank positive semidefinite matrix. The rich Riemannian geometry of the set of fixed-rank PSD matrices is exploited through a geometric optimization approach.

The resulting algorithms overcome the main difficulties encountered by the previously proposed methods as they scale to high-dimensional problems, and they naturally enforce the rank constraint as well as the positive definite property while leaving the range space of the matrix free to evolve during optimization.

We apply the proposed algorithms to the problem of learning a distance function from data, when the distance is parameterized by a fixed-rank positive semidefinite matrix. The good performance of the proposed algorithms is illustrated over several benchmarks.

Acknowledgments

This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. The scientific responsibility rests with its authors. Gilles Meyer is supported as an FRS-FNRS research fellow (Belgian Fund for Scientific Research).

Appendix A. Convergence Proof of Algorithm (9)

Bottou (1998) reviews the mathematical tools required to prove almost sure convergence, that is asymptotic convergence with probability one, of stochastic gradient algorithms. Almost sure convergence follows from the following five assumptions:

(A1) $F(\mathbf{G}) = \mathbb{E}_{\mathbf{X},y}\{\ell(\hat{y},y)\} \geq 0$ is three times differentiable with bounded derivatives,

(A2) the step sizes satisfy $\sum_{t=1}^{\infty} \eta_t^2 < \infty$ and $\sum_{t=1}^{\infty} \eta_t = \infty$,

(A3) $\mathbb{E}_{\mathbf{X},y}\{\|\mathrm{grad}\, f(\mathbf{G})\|_F^2\} \leq k_1 + k_2\|\mathbf{G}\|_F^2$, where $f(\mathbf{G}) = \ell(\hat{y},y)$,

(A4) $\exists h_1 > 0, \inf_{\|\mathbf{G}\|_F^2 > h_1} \mathrm{Tr}(\mathbf{G}^T \mathbb{E}_{\mathbf{X},y}\{\mathrm{grad}\, f(\mathbf{G})\}) > 0$,

(A5) $\exists h_2 > h_1, \forall (\mathbf{X},y) \in \mathcal{X} \times \mathcal{Y}, \sup_{\|\mathbf{G}\|_F^2 < h_2} \|\mathrm{grad}\, f(\mathbf{G})\|_F \leq k_3$,

where $\|\cdot\|_F$ is the Frobenius norm. Provided that algorithm (9) is equipped with an adaptive step size $s_t = \eta_t / \max(\|\mathbf{G}_t\|_F^2, 1)$, where η_t satisfy (A2), we have the following convergence result.

Proposition 1 *For bounded data* $(\mathbf{X},y)$, *algorithm* (9) *equipped with the step size s_t defined above converges almost surely to the set of stationary points of the cost function* $\mathbb{E}_{\mathbf{X},y}\{(\hat{y}-y)^2/2\}$.

Proof The proof is completed in two steps. First, it is shown that the stochastic sequence

$$u_t = \max(h_2, \|\mathbf{G}_t\|_F^2),$$

defines a Lyapunov process (always positive and decreasing on average) which is bounded almost surely by h_2. This implies that $\mathbf{G}_t$ is almost surely confined within distance $\sqrt{h_2}$ from the origin and provides almost sure bounds on all continuous functions of $\mathbf{G}_t$. In Bottou (1998), confinement is essentially based on (A3) and (A4). In the current proof, we rely on the fact that $\mathbb{E}_{\mathbf{X},y}\{\|\mathrm{grad}\, f(\mathbf{G})/\max(\|\mathbf{G}\|_F^2, 1)\|_F^2\} \leq k_1 + k_2\|\mathbf{G}\|_F^2$.

Second, the Lyapunov process $v_t = F(\mathbf{G}_t) \geq 0$ is proved to converge almost surely. Convergence of $F(\mathbf{G}_t)$ is then used to show that $w_t = \mathrm{grad}\ F(\mathbf{G}_t)$ tends to zero almost surely. Technical details are adapted from the paper of Bottou (1998). ■

In practice, saddle points and local maxima are unstable solutions while convergence to asymptotic plateaus is excluded by (A4). As a result, almost sure convergence to a local minimum of the expected cost is obtained.

References

P.-A. Absil, R. Mahony, and R. Sepulchre. Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. *Acta Appl. Math.*, 80(2):199–220, 2004.

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

R. Arora. On learning rotations. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22, pages 55–63. MIT Press, 2009.

V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 29 (1):328–347, 2007.

A. Asuncion and D.J. Newman. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`. University of California, Irvine, School of Information and Computer Sciences.

F. Bach and M. I. Jordan. Predictive low-rank decomposition for kernel methods. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.

S. Bonnabel and R. Sepulchre. Riemannian metric and geometric mean for positive semidefinite matrices of fixed rank. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1055–1070, 2009.

I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, second edition, 2005.

L. Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, 1998.

L. Bottou. Stochastic learning. In *Advanced Lectures on Machine Learning*, number 3176 in Lecture Notes in Artificial Intelligence, LNAI-3176, pages 146–168. Springer Verlag, 2004.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. MIT Press, 2007.

D. Cai, X. He, and J. Han. Efficient kernel discriminant analysis via spectral regression. In *Proceedings of the International Conference on Data Mining (ICDM07)*, 2007.

R. Chatpatanasiri, T. Korsrilabutr, P. Tangchanachaianan, and B. Kijsirikul. A new kernelization framework for Mahalanobis distance learning algorithms. *Neurocomputing*, 73(10-12):1570–1579, 2010.

T. Chen, Y. Hua, and W.-Y. Yan. Global convergence of Oja's subspace algorithm for principal component extraction. *IEEE Transaction Neural Networks*, 9(1):58–67, 1998.

T. F. Cox and M.A.A. Cox. *Multidimensional Scaling.* Chapman and Hall, 2001.

K. Crammer. Online tracking of linear subspaces. In *Proceedings of 19th Annual Conference on Learning Theory (COLT)*, 2006.

J. V. Davis and I. S. Dhillon. Structured metric learning for high dimensional problems. In *Proceedings of the 14th ACM SIGKDD conference on Knowledge Discovery and Data Mining*, 2008.

J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007.

A. Edelman, T.A. Arias, and S.T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

J. Faraut and A. Koranyi. *Analysis on Symmetric Cones*. Oxford University Press, 1994.

S. Fine, K. Scheinberg, N. Cristianini, J. Shawe-Taylor, and B. Williamson. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

A. Globerson and S. Roweis. Metric learning by collapsing classes. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18, pages 451–458. MIT Press, 2005.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17, pages 513–520. MIT Press, 2004.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman. Online metric learning and fast similarity search. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21, pages 761–768. MIT Press, 2008.

P. Jain, B. Kulis, and I. S. Dhillon. Inductive regularized learning of kernel functions. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23, pages 946–954. MIT Press, 2010.

M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization for semidefinite convex problems. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

J. Kivinen and M.K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Journal of Information and Computation*, 132(1):1–64, January 1997.

B. Kulis, M. Sustik, and I. S. Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 10:341–376, 2009.

J. Kwok and I. Tsang. Learning with idealized kernels. *Proceedings of the 20th International Conference on Machine learning (ICML)*, 2003.

G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In David Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan Kaufman, 1989.

Y. Matsuda and K. Yamaguchi. Global mapping analysis: stochastic gradient algorithm in SSTRESS and classical MDS stress. In *Proceedings of International Conference on Neural Information Processing*, 2001.

J. Nocedal and S. J. Wright. *Numerical Optimization, Second Edition*. Springer, 2006.

E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5: 927 – 935, 1992.

E.F. Petricoin, D.K. Ornstein, C.P. Paweletz, A.M. Ardekani, P.S. Hackett, B.A. Hitt, A. Velassco, C. Trucco, L. Wiegand, K. Wood, C.B. Simone, P.J. Levine, W.M. Linehan, M.R. Emmert-Buck, S.M. Steinberg, E.C Kohn, and L.A. Liotta. Serum proteomic patterns for detection of prostate cancer. *Journal of the National Cancer Institute*, 94(20):1576–1578, 2002.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(3):1299–1319, 1998.

S. Shalev-Shwartz, Y. Singer, and A. Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

S.T. Smith. Covariance, subspace, and intrinsic Cramér-Rao bounds. *IEEE Transactions on Signal Processing*, 53(5):1610–1630, 2005.

L. Song, A. Smola, K. Borgwardt, and A. Gretton. Colored maximum variance unfolding. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 1385–1392. MIT Press, 2007.

A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI)*, 2000.

L. Torresani and K. Lee. Large margin component analysis. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 1385–1392. MIT Press, 2006.

K. Tsuda, G. Ratsch, and M. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *Journal of Machine Learning Research*, 6:995–1018, 2005.

M. Warmuth. Winnowing subspaces. *Proceedings of the 24th international conference on Machine learning (ICML)*, 2007.

K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

K. Weinberger, F. Sha, and L. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pages 839–846, 2004.

E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 505–512. MIT Press, 2002.

L. Yang. Distance metric learning: a comprehensive survey. Technical report, Michigan State University, 2006.

J. Zhuang, I. Tsang, and S. Hoi. SimpleNPKL: simple non-parametric kernel learning. *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.