The Application of an Oblique-Projected Landweber Method to a Model of Supervised Learning

Björn Johansson, Tommy Elfving, Vladimir Kozlov, Yair Censor, and Gösta Granlund

September 6, 2004

1Computer Vision Laboratory, Dept. of Electrical Engineering, Linköping University, SE-581 83, Linköping, Sweden, {bjorn,gosta}@isy.liu.se
2Numerical Analysis Division, Dept. of Mathematics, Linköping University, SE-581 83, Linköping, Sweden, toelf@mai.liu.se
3Applied Mathematics Division, Dept. of Mathematics, Linköping University, SE-581 83, Linköping, Sweden, vlkoz@mai.liu.se
4Department of Mathematics, University of Haifa, Mt. Carmel, Haifa 31905, Israel, yair@math.haifa.ac.il

Abstract

This report brings together a novel approach to some computer vision problems and a particular algorithmic development of the Landweber iterative algorithm. The algorithm solves a class of high-dimensional, sparse, and constrained least-squares problems, which arise in various computer vision learning tasks, such as object recognition and object pose estimation. The algorithm has recently been applied to these problems, but it has been used rather heuristically. In this report we describe the method and put it on firm mathematical ground. We consider a convexly constrained weighted least-squares problem and propose for its solution a projected Landweber method which employs oblique projections onto the closed convex constraint set. We formulate the problem, present the algorithm and
work out its convergence properties, including a rate-of-convergence result. The results are put in perspective of currently available projected Landweber methods. The application to supervised learning is described, and the method is evaluated in a function approximation experiment.

**Keywords** Projected Landweber, preconditioner, nonnegative constraint, supervised learning, channel representation

### 1 Introduction

The aim of this report is to describe a novel approach to the modeling of computer vision problems. The model rests on principles of supervised learning and its solution employs an oblique-projected Landweber method. It is the *combination* of the algorithmic design, its validation, the applied field to which it fits and the experimental work — the *combination* of all these IS the scientific goal of the present report.

Computer vision problems present many logical and computational challenges and are extremely important for many real-world applications. Often the computer vision problem gives rise to function (or other mathematical relations) approximation problems, which can be solved by constructing a model of the approximated function and detecting the correct values of the model’s parameters by a supervised learning process. Such a supervised learning process exposes the model to a number of known functions according to which the model adjusts (“learns”) its parameters. These known functions constitute the training set for the supervised learning process and when this learning process is complete the model is confronted with the practical function approximation problems.

The resulting function approximation problems are quite complex and typically computationally demanding. A tool for handling function approximation problems, within the approach described above, is to use the concept of *channel representation*. The purpose of this concept is to represent a vector (namely, a point) \( z \in \mathbb{R}^M \) by some other vector such that there will be an advantage, for the application at hand where \( z \) plays a role, in the new representation. The channel representation, discussed here, performs a mapping of data into a higher-dimensional space. While this would seem, on the face of it, to be undesirable it does enable us to introduce locality in the representation, while retaining excellent interpolation properties. The channel representation concept has a number of additional important features that will be discussed later in this report.
Coming to the algorithmic side, we describe in this work the application of a convexly-constrained, weighted, oblique-projected Landweber algorithm to high-dimensional, sparse and constrained least-squares problems that arise in our approach to computer vision, see, e.g., [Björck, 1996, Section 7.7.2] for a basic Landweber formula. In the literature of optimization theory projected Landweber methods are referred to as projected gradient methods, see, e.g., [Bertsekas, 1999, Chapter 2] or [Levitin and Polyak, 1968], where similar methods are treated. We present an independent analysis of the method that we use in order to make this report self-contained and to emphasize the details in which our analysis slightly modify and extend known results. To put our work in perspective, we discuss the algorithm, at the end of Sections 2 and 3, in the light of a recent state-of-the-art analysis by [Piana and Bertero, 1997, Section 2]. The algorithm handles convex constraints by allowing oblique projections onto the convex sets, rather than standard orthogonal (least-Euclidean-distance) projections (all these terms are clarified in the next section). Furthermore, the method contains a user-specified diagonal weighting matrix which allows component-wise weighting. These features render this scheme extra implementational flexibility.

We present a convergence proof for the general case in Section 2. Although convergence can also be concluded in various ways from the works of Eicke [1992], Piana and Bertero [1997], and Bertsekas [1999], we find it useful to present a complete and streamlined proof in finite dimensions (thus avoiding much of the machinery needed in infinite dimensions). We will discuss our proof and its relation with results from the above references at the end of Section 2. In Section 3 we narrow the discussion to the mathematically simpler but practically very important case of nonnegativity constraints. We first motivate the case of nonnegativity as a tool for obtaining smooth solutions. Then we sharpen our analysis and reach a linear rate-of-convergence result. Here we do not assume full rank of the matrix, as, e.g., assumed in Bertsekas [1999]. Section 4 describes our work on function approximation via supervised learning, aimed at computer vision problems, and shows how the use of the algorithms presented here enables us to extend significantly the horizon of our experimental work in this field. Computer vision often involves high-dimensional signal processing tasks which put a high demand on the function modeling and on the learning strategy that is employed to compute the model parameters. It is, therefore, desirable that the model and the learning algorithm are computationally efficient and that they require modest memory storage. The model and learning strategy presented in [Granlund et al., 2002; Granlund, 2000] are an attempt to meet these demands. In particular, the learning method of [Granlund et al., 2002] is a simple, yet efficient, iterative algorithm which has been
initially used rather heuristically and is put here on firm mathematical ground.

2 The Convexly-Constrained, Weighted,
Oblique-Projected Landweber Method

In this section we consider a convexly constrained weighted least-squares problem and propose for its solution a projected Landweber method which employs oblique projections onto the closed convex constraint set. Below we formulate the problem, present the algorithm and work out its convergence analysis. These steps are followed by a discussion of how this algorithm differs from currently available projected Landweber methods for this problem.

Let $R^m$ and $R^n$ denote the $m$-dimensional and $n$-dimensional Euclidean spaces, respectively. Further, let $\langle \cdot, \cdot \rangle$ denote the Euclidean inner product, and $\| \cdot \|$ - the induced norm. We also supply $R^m$ and $R^n$ with weighted inner products and norms. Given a symmetric and positive definite matrix $\Lambda$ of order $m$ or $n$, respectively, we denote by $\langle \cdot, \cdot \rangle_\Lambda := \langle \cdot, \Lambda \cdot \rangle$ the $\Lambda$-inner product, and by $\| \cdot \|_\Lambda$ the induced weighted norm.

Let $A = (a_{ij})_{i,j=1}^{m,n}$ be an $m \times n$ real matrix, let $W$ be a real symmetric and positive definite matrix of order $m$, and let $b = (b_i)_{i=1}^m \in R^m$. We define, for any $x \in R^n$, the function

$$e(x) := \frac{1}{2} \|Ax - b\|^2_W = \frac{1}{2} \langle (Ax - b), W(Ax - b) \rangle. \tag{1}$$

Moreover, let $\Omega \subseteq R^n$ be a given nonempty, closed convex set. The weighted, oblique-projected Landweber method that we propose and study here is designed to solve the convexly-constrained least-squares problem

$$\min \{ e(x) \mid x \in \Omega \}. \tag{2}$$

We need the following additional notations. Let $D$ be a real symmetric and positive definite matrix of order $n$, and let $D^{-1}$ denote the inverse of $D$. The oblique projection with respect to $D^{-1}$, of a vector $y \in R^n$ onto the set $\Omega$ is defined as the point $z \in \Omega$ for which

$$z = P_\Omega(y) := \arg \min \{ \|u - y\|_{D^{-1}} \mid u \in \Omega \}. \tag{3}$$
For $D = I$, the unit matrix, $P_\Omega(y)$ becomes the orthogonal projection of $y$ onto the set $\Omega$, and the theorem that guarantees the existence and uniqueness of orthogonal projections onto a closed and convex set, see, e.g., [Hiriart-Urruty and Lemaréchal, 2001, Chapter A, Section 3], can be easily revised to cover oblique projections. Note that if $\Omega = \{x \in \mathbb{R}^n \mid x \geq 0\} = \mathbb{R}_+^n$ (the nonnegative orthant) and $D$ is diagonal, then

$$P_\Omega(y) = \max(y, 0),$$

where $\max(u, v)$ denotes the component-wise maximum of $u, v \in \mathbb{R}^n$.

**Algorithm 1** (Projected Landweber’s Method with Oblique Projections and Weighting).

*Initialization:* $x^0 \in \Omega$ is arbitrary.

*Iterative step:*

$$x^{k+1} = P_\Omega(x^k - \gamma D\nabla e(x^k)), \text{ for all } k = 0, 1, \ldots$$

where $\nabla$ is the gradient, i.e.,

$$\nabla e(x) = A^T W (Ax - b),$$

and $\gamma$ is a positive relaxation parameter whose range of permissible values is specified below.

When the matrix $D$ is diagonal we call the method a component-wise-weighted method. In practice this is an important special case, since for box-constraints it simplifies the actual computation of the oblique projection. Our convergence result for Algorithm 1 is Theorem 6 which states that any sequence $\{x^k\}_{k \geq 0}$, generated by the algorithm, converges to a solution of problem (2). The proof relies on four auxiliary propositions that we now present. For notational convenience we define the operator

$$T(x) := P_\Omega(x - \gamma D\nabla e(x)).$$

**Proposition 2** Assume that $\gamma > 0$. If $\{x^k\}_{k \geq 0}$ is generated by Algorithm 1 then, for any $k \geq 0$,

$$\gamma \langle (T(x^k) - x^k), \nabla e(x^k) \rangle \leq - \langle (x^k - T(x^k)), D^{-1}(x^k - T(x^k)) \rangle.$$
Proof. A necessary and sufficient condition for (3) to hold is
\[
\langle (v - P_\Omega(y)), D^{-1}(y - P_\Omega(y)) \rangle \leq 0, \quad \text{for all } v \in \Omega, \quad (9)
\]
see, e.g., [Hiriart-Urruty and Lemaréchal, 2001, Theorem 3.1.1] for the orthogonal projections case, which can be easily extended to oblique projections (or apply Theorem 2.4.2 from [Censor and Zenios, 1997] with the choice of the function \( f \) in that theorem as \( f(x) = \frac{1}{2} \|x\|_{D^{-1}}^2 \)). Choosing \( y = x^k - \gamma D\nabla e(x^k) \) and \( v = x^k \) \( (v \in \Omega \) because \( x^k \) is the result of an earlier step which projected on \( \Omega \)) we obtain
\[
\langle (T(x^k) - x^k), D^{-1}(x^k - \gamma D\nabla e(x^k) - T(x^k)) \rangle \leq 0, \quad (10)
\]
from which (8) follows.

In the sequel we denote by \( \rho(Q) \) the spectral radius, i.e., the largest, in absolute value, eigenvalue of the matrix \( Q \), by \( \lambda_{\min}(C) \) the smallest eigenvalue of the matrix \( C \), by \( \lambda(C) \) any eigenvalue of \( C \), and by \( A^T \) the transpose of the matrix \( A \). Further \( D^{1/2} \) is the square root of a positive definite and symmetric matrix \( D \).

Proposition 3 Let \( Q := D^{1/2}A^TWA \) and assume that \( 0 < \gamma < 2/\rho(Q) \). Then there exists a constant \( c_1 > 0 \) such that for every sequence \( \{x^k\}_{k \geq 0} \), generated by Algorithm 1, we have
\[
e(x^{k+1}) \leq e(x^k) - c_1\|x^{k+1} - x^k\|_{D^{-1}}^2. \quad (11)
\]

Proof. By the mean value theorem we have
\[
e(x + y) = e(x) + \int_0^1 \langle y, \nabla e(x + ty) \rangle dt. \quad (12)
\]
Then, using (6), we get
\[
e(x + y) = e(x) + \int_0^1 \langle y, (A^TWAx + tA^TWAy - A^TWb) \rangle dt
\]
\[
= e(x) + \langle y, A^TWAx \rangle + \frac{1}{2} \langle y, A^TWAy \rangle - \langle y, A^TWb \rangle
\]
\[
= e(x) + \langle y, A^TWA(x - b) \rangle + \frac{1}{2} \langle y, A^TWAy \rangle. \quad (13)
\]
Substituting \( y = x^{k+1} - x^k \) and \( x = x^k \) we further get
\[
e(x^{k+1}) = e(x^k) + \langle (x^{k+1} - x^k), A^TWA(x^{k+1} - x^k) \rangle
\]
\[
+ \frac{1}{2} \langle (x^{k+1} - x^k), A^TWA(x^{k+1} - x^k) \rangle. \quad (14)
\]
By (8), we have for the second term in the right-hand side of the last equation

\[ \langle (x^{k+1} - x^k), A^T W (Ax^k - b) \rangle \leq -\frac{1}{\gamma} \langle (x^{k+1} - x^k), D^{-1} (x^{k+1} - x^k) \rangle, \quad (15) \]

hence,

\[
e(x^{k+1}) \leq e(x^k) \]

\[
- \langle (x^{k+1} - x^k), D^{-1/2} (\frac{1}{\gamma} I - \frac{1}{2} D^{1/2} A^T W AD^{-1/2}) D^{-1/2} (x^{k+1} - x^k) \rangle. \quad (16)
\]

Defining the matrix

\[
C := \frac{1}{\gamma} I - \frac{1}{2} D^{1/2} A^T W AD^{-1/2} = \frac{1}{\gamma} I - \frac{1}{2} Q, \quad (17)
\]

we have \( \langle x, D^{-1/2} C D^{-1/2} x \rangle \geq \lambda_{\text{min}}(C) \|x\|_D^{-1} \), for all \( x \in \mathbb{R}^n \), and

\[
e(x^{k+1}) \leq e(x^k) - \langle (x^{k+1} - x^k), D^{-1/2} C D^{-1/2} (x^{k+1} - x^k) \rangle \]

\[
\leq e(x^k) - \lambda_{\text{min}}(C) \|x^{k+1} - x^k\|_D^{-1}. \quad (18)
\]

Now from (17) and from the assumption on \( \gamma \),

\[
\lambda(C) = \frac{1}{\gamma} - \frac{1}{2} \lambda(Q) \quad \text{and} \quad 0 < \gamma < 2/\rho(Q), \quad (19)
\]

hence

\[
\lambda_{\text{min}}(C) = \frac{1}{\gamma} - \frac{1}{2} \rho(Q) > \frac{\rho(Q)}{2} - \frac{1}{2} \rho(Q) = 0. \quad (20)
\]

So, the result holds with \( c_1 := \lambda_{\text{min}}(C) > 0. \)

**Proposition 4** Assume that \( \gamma > 0 \). Then the following two conditions are equivalent

\[
x^* \in \arg\min\{e(x) \mid x \in \Omega\}, \quad (21)
\]

\[
x^* = P_\Omega (x^* - \gamma D\nabla e(x^*)) \equiv T(x^*). \quad (22)
\]
Proof. By (3), the point $x^*$ fulfills (22) if and only if

$$x^* = \text{argmin}\{\|x - x^* + \gamma D\nabla e(x^*)\|_{D^{-1}} \mid x \in \Omega\},$$

(23)

which is equivalent to $x^* \in \text{argmin}\{f(x) \mid x \in \Omega\}$ with the function $f(x) := \frac{1}{2}\|x - x^* + \gamma D\nabla e(x^*)\|_{D^{-1}}^2$. From standard optimality conditions it is well-known, since $f(x)$ is continuous and convex, that this is equivalent to

$$\langle x - x^*, \nabla f(x^*) \rangle \geq 0, \text{ for all } x \in \Omega.$$  

(24)

The gradient of $f(x)$ is

$$\nabla f(x) = D^{-1}(x - x^* + \gamma D\nabla e(x^*)),$$  

(25)

so,

$$\nabla f(x^*) = \gamma D^{-1} D\nabla e(x^*) = \gamma \nabla e(x^*),$$  

(26)

which, together with (24), gives

$$\langle x - x^*, \gamma \nabla e(x^*) \rangle \geq 0, \text{ for all } x \in \Omega.$$  

(27)

Since $\gamma > 0$ and $e(x)$ is continuous and convex, the equivalence with (21) follows.

Next we show Fejér-monotonicity, see, e.g., [Censor and Zenios, 1997, Definition 5.3.1], in the oblique norm sense, of any sequence generated by Algorithm 1, with respect to the solution set (21).

**Proposition 5** Under the assumptions of Proposition 3, any sequence of vectors \( \{x^k\}_{k \geq 0} \) generated by Algorithm 1, has the property

$$\|x^{k+1} - x^*\|_{D^{-1}} \leq \|x^k - x^*\|_{D^{-1}}, \text{ for all } k \geq 0,$$

(28)

for any $x^* \in \text{argmin}\{e(x) \mid x \in \Omega\}$.

**Proof.** Using the non-expansiveness of the projection operator (see, e.g., [Hiriart-Urruty and Lemaréchal, 2001, page 48]) we get

$$\|x^{k+1} - x^*\|_{D^{-1}} = \|P_{\Omega}(x^k - \gamma D\nabla e(x^k)) - P_{\Omega}(x^* - \gamma D\nabla e(x^*))\|_{D^{-1}}$$

$$\leq \|(x^k - x^*) - \gamma D(\nabla e(x^k) - \nabla e(x^*))\|_{D^{-1}}$$

$$= \|(I - \gamma DA^TWA)(x^k - x^*)\|_{D^{-1}}$$

$$\equiv \|V(x^k - x^*)\|_{D^{-1}},$$

(29)
where
\[ V = I - \gamma DA^TWA. \] 
(30)

Now for any \( u \in \mathbb{R}^n \) we have
\[
\|Vu\|_{D^{-1}}^2 = \langle Vu, D^{-1}Vu \rangle \\
= \langle VD^{1/2}D^{-1/2}u, D^{-1}VD^{1/2}D^{-1/2}u \rangle \\
\leq \rho(D^{1/2}V^TD^{-1/2}VD^{1/2})\|u\|_{D^{-1}}^2.
\] 
(31)

Abbreviating \( Q_1 := D^{1/2}V^TD^{-1/2}VD^{1/2} \) we obtain from (29) and (31)
\[
\|x^{k+1} - x^*\|_{D^{-1}} \leq \sqrt{\rho(Q_1)}\|x^k - x^*\|_{D^{-1}}.
\] 
(32)

Using (30) we arrive at
\[
Q_1 = D^{1/2}V^TD^{-1/2}VD^{1/2} \\
= D^{1/2}(I - \gamma A^TWAD)D^{-1/2}D^{-1/2}(I - \gamma DA^TWA)D^{1/2} \\
= (I - \gamma D^{1/2}A^TWAD^{1/2})(I - \gamma D^{1/2}A^TWA^D^{1/2}),
\] 
(33)

so that \( \sqrt{\rho(Q_1)} = \rho(I - \gamma D^{1/2}A^TWA^D^{1/2}) \). The assumption \( 0 < \gamma < 2/\rho(Q) \) then guarantees that \( \sqrt{\rho(Q_1)} \leq 1 \) and the conclusion follows from (32).

With the last four propositions in hand we are ready to prove convergence of Algorithm 1.

**Theorem 6** Assume that \( 0 < \gamma < 2/\rho(Q) \) where \( Q = D^{1/2}A^TWA^D^{1/2} \). Then any sequence \( \{x^k\}_{k \geq 0} \), generated by Algorithm 1, converges to a solution of the problem (2).

**Proof.** A Fejér-monotone sequence is always bounded (see, e.g., [Censor and Zenios, 1997, page 84]), therefore, we know from Proposition 5 that \( \{x^k\}_{k \geq 0} \) is bounded, thus it has at least one cluster point. The sequence \( \{e(x^k)\}_{k \geq 0} \) is bounded from below by zero, since \( e(x) \geq 0 \), and is monotonically decreasing, by Proposition 3. Therefore, \( \{e(x^k)\}_{k \geq 0} \) converges, hence, by (11),
\[
\lim_{k \to \infty} \|x^{k+1} - x^k\|_{D^{-1}} = \lim_{k \to \infty} \|T(x^k) - x^k\|_{D^{-1}} = 0.
\] 
(34)
The operator $T(\cdot)$ is continuous (by the non-expansiveness property of the projection operator), hence every cluster point $\hat{x}$ of $\{x^k\}_{k \geq 0}$ satisfies
\[ \hat{x} = T(\hat{x}), \] (35)
and thus, by proposition 4, $\hat{x}$ is a solution of problem (2).

Finally, if $\bar{x}$ is any cluster point of $\{x^k\}_{k \geq 0}$, then $\bar{x}$ solves (2) and Proposition 5 guarantees that, for all $k \geq 0$,
\[ 0 \leq \|\bar{x} - x^{k+1}\|_{D^{-1}} \leq \|\bar{x} - x^k\|_{D^{-1}}. \] (36)
Thus, $\lim_{k \to \infty} \|\bar{x} - x^k\|_{D^{-1}}$ exists and since $\bar{x}$ is a cluster point, this limit must be zero, proving that $\bar{x}$ is the limit of $\{x^k\}_{k \geq 0}$. ■

We conclude this section by accurately putting our algorithm in context with currently available Landweber algorithmic schemes. We present and compare our algorithm with convergence results from the literature, in particular the description in the state of the art paper Piana and Bertero [1997]. First we observe, combining (5) and (6), that
\[ x^{k+1} = P_\Omega(x^k - \gamma DA^T W(Ax^k - b)). \] (37)
In Piana and Bertero [1997] $A : X \to Y$ is a linear continuous operator and $X, Y$ are Hilbert spaces. Their method (2.9) reads (using our notation)
\[ x^{k+1} = P_\Omega(x^k - \gamma A^* (Ax^k - b)), \] (38)
where now $P_\Omega$ denotes the orthogonal projection. They state, using results of Eicke [1992] and others, in their Theorem 2.2 that for $0 < \gamma < 2/\|A\|^2$ the method converges weakly to a solution of problem (2). We next show that this result contains ours. Let $X = \mathcal{R}^n$, $Y = \mathcal{R}^m$ with scalar products $\langle , \rangle_X = x^T D^{-1} x$ and $\langle , \rangle_Y = y^T W y$. It easily follows that the adjoint operator becomes $A^* = DA^T W$. Further, since
\[ \|Ax\|_Y^2 \|x\|_X^2 = x^T A^T W Ax / X^T D^{-1} x = z^T Qz / z^T z, \] (39)
where $Q = D^{1/2} A^T W A D^{1/2}$ it follows that $\|A\|^2 = \|Q\|$. Note also by this choice of spaces that $P_\Omega$ is the oblique projection as used in our algorithm. It follows that our Theorem 6 is a special case of Theorem 2.2 in Piana and Bertero [1997], the convergence now being in norm. In Section 4 of Piana and Bertero [1997] the authors consider a related method (4.1) to Algorithm 1. In this related method $W = I$ and $D$ is an operator commuting with the operator $A^* A$. 

10
We next discuss the work by Bertsekas as summarized in Bertsekas [1999], Section 2.3. Bertsekas discusses the scaled gradient projection method (SGPM) on pp. 229–230. In Exercise 2.3.1 (p. 241) the reader is asked to verify that the SGPM can be rewritten using oblique projections and we then retrieve Algorithm 1 above (using $W = I$). In Proposition 2.3.4 it is shown that for two specific choices of stepsize (the limited minimization rule and the Armijo rule) every limit point is stationary. This means that every limit point satisfies the optimality conditions (Proposition 2.1.2, p. 194). Note that Bertsekas does not include our case, i.e. a constant stepsize. This case is considered for the gradient projection method (Proposition 2.3.2, p. 235).

Automatically we number all equations displayed. It is instructive to have a complete and streamlined convergence proof, as we do here. Since we are only considering finite dimensions we also avoid much of the machinery needed for the infinite case cf. Eicke [1992].

3 Nonnegativity Constrained Problem

The special case of nonnegativity constraints is of great practical importance. We deepen our analysis for this case regarding regularization properties and rate-of-convergence properties.

3.1 Regularization Property of the Nonnegativity Constrained Problem

It is, besides from experimental support, difficult to analyze the regularization properties of the nonnegativity constraint. But it is at least possible to derive an upper limit to the solution. Let $x^*$ be a solution to

$$
\min \{ \|Ax - b\| \mid x \geq 0 \}, \tag{40}
$$

and let $x^\circ = (x_j^\circ)$ be a vector such that

$$
x_j^\circ = \arg \min \{ \|a_j^T x - b\| \mid x_j \geq 0 \} = \frac{\langle a_j, b \rangle}{\langle a_j, a_j \rangle} \tag{41}
$$

where the vector $a_j$ denotes the $j$-th column in $A$. Thus, $x^\circ$ is the solution when each element $x_j$ is optimized separately.
Theorem 7 If $A \geq 0$ element-wise, and every column in $A$ has at least one nonzero element, i.e., $a^j \neq 0$ for all $j$, then $0 \leq x^* \leq x^\circ$.

Proof. Assume that we have a vector $x$, such that $x_j - x^\circ_j > 0$ for some $j$. Let
\[
y = x - (x_j - x^\circ_j)e^j,
\]
where $e^j$ is the $j$-th unit vector. Since both $A$ and $y$ are nonnegative, and since $y_j = x^\circ_j$, it follows that, by 41,
\[
\langle a^j, Ay \rangle = \sum_{t=1}^n \langle a^j, a^t \rangle y_t \geq \langle a^j, a^j \rangle y_j = \langle a^j, a^j \rangle x^\circ_j = \langle a^j, b \rangle.
\]
Therefore,
\[
\|Ax - b\|^2 - \|Ay - b\|^2 = \|Ay - b + (x_j - x^\circ_j)a^j\|^2 - \|Ay - b\|^2
\]
\[
= (x_j - x^\circ_j)^2 \langle a^j, a^j \rangle + 2(x_j - x^\circ_j) \langle a^j, Ay - b \rangle
\]
\[
\geq (x_j - x^\circ_j)^2 \langle a^j, a^j \rangle
\]
\[
> 0.
\]
Hence, $x$ cannot be a solution to (40), and the proof follows. 

Theorem 7 states that the solution $x^*$ is smaller than the solution to the problem of optimizing each element $x_j$ separately. Note that the theorem does not hold for the unconstrained problem, where the elements of the solution $x^*$ can assume very large values for ill-conditioned problems.

3.2 Rate of Convergence for the Nonnegativity Constrained Problem

In this section we focus on the special case in which the constraint set of problem (2) is the nonnegative orthant $\Omega = R^n_\geq := \{x \in R^n \mid x \geq 0\}$. This very important case in practical applications admits a richer mathematical analysis which allows us to estimate the rate of convergence of any sequence generated by Algorithm 1 for $\Omega = R^n_\geq$. In our knowledge this estimate is new. The main difference, in comparison with a similar result in [Bertsekas, 1999, page 229], is that we do not suppose here that the matrix $A$ is of full rank. Bertsekas and Gafni [1982] showed linear rate of convergence of the gradient projected method (i.e., the projected
Landweber iteration) under the assumption that, in our case, the matrix $A$ has full rank, so that the smallest eigenvalue of $A^T A$ is larger than zero. Further generalizations can be found, e.g., in the paper by Luo and Tseng [1993], but using the same condition on the matrix $A$ (c.f. [Luo and Tseng, 1993, eq. (1.4)]).

We will need the following additional condition.

**Condition 8** Let $A \geq 0$, i.e., $a_{ij} \geq 0$ for all $i$ and all $j$, and such that $\sum_{j=1}^{n} a_{ij} > 0$ for all $i$, i.e., every row has at least one nonzero element. Further, let $W = \text{diag}(w_1, w_2, \ldots, w_m)$, and let $D = \text{diag}(d_1, d_2, \ldots, d_n)$ be real positive diagonal matrices, i.e., $w_j, d_j > 0$, for all $j$.

For the treatment given below we rewrite Algorithm 1 in this case as follows.

**Algorithm 9** *(Algorithm for the Nonnegativity Constrained Problem).*

**Initialization:** $x^0, y^0 \in R^n_\geq$ is arbitrary.

**Iterative step:** Given $x^k$ calculate:

$$
\begin{align*}
    y^{k+1} &= x^k - D A^T W (A x^k - b), \\
    x^{k+1} &= \max(y^{k+1}, 0).
\end{align*}
$$

(45)

In the next theorem we present the result on the rate of convergence for Algorithm 9 which is the oblique-projected component-wise-weighted Landweber method (Algorithm 1) when $\Omega = R^n_\geq$ and $\gamma$ is included in $D$. Define

$$
B := D A^T W A \quad \text{and} \quad g := D A^T W b.
$$

(46)

Note that $B$ is related to $Q$ in Theorem 6 by $B = D^{1/2} Q D^{-1/2}$ and, consequently, $ho(B) = \rho(Q)$.

**Theorem 10** Assume that $A$, $b$, $W$, and $D$ fulfill Condition 8, and that $\rho(B) < 2$. Then the sequences $\{x^k\}_{k\geq0}$ and $\{y^k\}_{k\geq0}$, generated by Algorithm 9, are convergent, and $\{x^k\}_{k\geq0}$ converges to a solution of problem (2) with $\Omega = R^n_\geq$.

Moreover, if $x^*$ and $y^*$ are the limits of the two sequences, respectively, then there exists a real $q \in (0, 1)$ such that, for all $k \geq 0$,

$$
\|x^k - x^*\|_{D^{-1}} \leq \frac{q^k}{1-q} \|g\|_{D^{-1}} \quad \text{and} \quad \|y^k - y^*\|_{D^{-1}} \leq \frac{q^k}{1-q} \|g\|_{D^{-1}}.
$$

(47)
Proof. It is easy to verify that the matrix $B$, defined in (46), is nonnegative and symmetric with respect to the $D^{-1}$-inner product in $\mathbb{R}^n$, i.e.
\[
\langle Bx, y \rangle_{D^{-1}} = \langle x, By \rangle_{D^{-1}}, \text{ for all } x, y \in \mathbb{R}^n.
\] (48)
Moreover, since $B$ is nonnegative and $\rho(B) < 2$ we have
\[
\|(I - B)x\|_{D^{-1}} \leq \|x\|_{D^{-1}}, \text{ for every } x \in \mathbb{R}^n.
\] (49)
We denote by $\mathcal{N}(A)$ the kernel (i.e., the null space) of the matrix $A$ and introduce the real number $\delta$
\[
\delta := 1 - \max \left\{ \frac{\|P_{\mathcal{N}(A)}(x)\|_{D^{-1}}}{\|x\|_{D^{-1}}} \mid x \in \mathbb{R}_0^n, x \neq 0 \right\},
\] (50)
where $P_{\mathcal{N}(A)}(x)$ is the oblique projection with respect to $D$ of $x$ onto $\mathcal{N}(A)$, as defined in (3). The assumptions on the matrix $A$ imply that the only element in $\mathcal{N}(A)$ with nonnegative components is zero, thus we must have $0 < \delta \leq 1$.

Let $J$ be a nonempty subset of $\{1, 2, \ldots, n\}$ and denote the family of all such subsets by $\mathcal{J}$. For each $J \in \mathcal{J}$ we introduce the subspace $X_J \subseteq \mathbb{R}^n$,
\[
X_J := \{ x \in \mathbb{R}^n \mid j \notin J \text{ implies } x_j = 0 \},
\] (51)
define $H_J := X_J \cap \mathcal{N}(A)$ and denote by $Y_J$ the subspace of $X_J$ which is $D^{-1}$-orthogonal to $H_J$. Given any $J \in \mathcal{J}$, we let $\sigma_J$ be the largest nonnegative number for which
\[
\|(I - B)y\|_{D^{-1}}^2 \leq (1 - \sigma_J)\|y\|_{D^{-1}}^2, \text{ for all } y \in Y_J,
\] (52)
where $I$ denotes the unit matrix. Clearly, $\sigma_J$ is actually positive. Rewriting the first line of (45) as
\[
y^{k+1} = x^k - (Bx^k - g),
\] (53)
we have
\[
y^{k+1} - y^k = (I - B)(x^k - x^{k-1}), \text{ for all } k \geq 0.
\] (54)
Furthermore, if $y_{j}^{k+1} \geq 0$ or $y_{j}^{k} \geq 0$ then
\[
|x_{j}^{k+1} - x_{j}^{k}| + |y_{j}^{k+1} - x_{j}^{k+1}| + |y_{j}^{k} - x_{j}^{k}| = |y_{j}^{k+1} - y_{j}^{k}|.
\] (55)
Since
\[\langle Bx - g, h \rangle_{D^{-1}} = \langle W(Ax - b), Ah \rangle = 0, \text{ for all } h \in \mathcal{N}(A) \]  
(56)

it follows that
\[\langle y^{k+1} - x^k, h \rangle_{D^{-1}} = 0, \text{ for all } h \in \mathcal{N}(A). \]  
(57)

Next we put \(w^k := x^k - x^{k-1}\) and define
\[J_k := \{j | 1 \leq j \leq n \text{ for which either } y_j^k \geq 0 \text{ or } y_j^{k-1} \geq 0\}. \]  
(58)

Since \(g\) is nonzero and contains only nonnegative components the set \(J_k\) is nonempty and we associate with it the subspace \(X_{J_k} \subseteq \mathbb{R}^n\). We decompose \(w^k\), which clearly belongs to \(X_{J_k}\), as \(w^k = w^{k,0} + w^{k,1}\) where \(w^{k,0} \in H_{J_k}\) and \(w^{k,1} \in Y_{J_k}\), and consider two cases.

Case I. \(\|w^{k,0}\|_{D^{-1}} \leq (1 - \varepsilon)\|w^k\|_{D^{-1}}\) for some \(\varepsilon \in (0, 1)\).

Regardless of the specific value of \(\varepsilon\), which will be chosen later, we have in this case
\[
\| (I - B)w^k \|_{D^{-1}}^2 = \| (I - B)w^{k,1} \|_{D^{-1}}^2 + \| (I - B)w^{k,0} \|_{D^{-1}}^2 \\
\leq (1 - \sigma_J)\|w^{k,1}\|_{D^{-1}}^2 + \|w^{k,0}\|_{D^{-1}}^2 \\
= (1 - \sigma_J)\|w^k\|_{D^{-1}}^2 + \sigma_J\|w^{k,0}\|_{D^{-1}}^2 \\
\leq (1 - \sigma_J)\|w^k\|_{D^{-1}}^2 + (1 - \varepsilon)^2 \sigma_J\|w^k\|_{D^{-1}}^2. 
\]  
(59)

Hence
\[
\| (I - B)w^k \|_{D^{-1}}^2 \leq (1 - \varepsilon \sigma_J)\|w^k\|_{D^{-1}}^2, 
\]  
(60)

which, together with (54) and (55), gives
\[
\|y^{k+1} - y^k\|_{D^{-1}} = \| (I - B)(x^{k+1} - x^k) \|_{D^{-1}} \\
\leq \sqrt{(1 - \varepsilon \sigma_J)}\|x^k - x^{k-1}\|_{D^{-1}} \\
\leq \sqrt{(1 - \varepsilon \sigma_J)}\|y^k - y^{k-1}\|_{D^{-1}}. 
\]  
(61)

Case II. \(\|w^{k,0}\|_{D^{-1}} \geq (1 - \varepsilon)\|w^k\|_{D^{-1}}\) for some \(\varepsilon \in (0, 1)\).

We rewrite \(w^k = y^k - x^{k-1} + x^k - y^k\) and assume that the components of \(y^k - x^{k-1}\) and of \(x^k - y^k\) with indices \(j \notin J_k\) are zero. For these new vectors we preserve the same notations. By (57) the vector \(y^k - x^{k-1}\) is \(D^{-1}\)-orthogonal to
Therefore, \( w^{k,0} = P_{H_{J_k}}(x^k - y^k) \) is the oblique projection of \( x^k - y^k \) onto \( H_{J_k} \), from which, by (50), we have

\[
(1 - \varepsilon)\|w^k\|_{D^{-1}} \leq \|w^{k,0}\|_{D^{-1}} = \|P_{H_{J_k}}(x^k - y^k)\|_{D^{-1}} \\
\leq (1 - \delta)\|x^k - y^k\|_{D^{-1}}.
\]

Therefore,

\[
\|w^k\|_{D^{-1}} \leq \frac{1 - \delta}{1 - \varepsilon}\|x^k - y^k\|_{D^{-1}},
\]

which, together with (55), implies

\[
\|w^k\|_{D^{-1}} \leq \frac{1 - \delta}{1 - \varepsilon}\|y^k - y^{k-1}\|_{D^{-1}}. \tag{62}
\]

Hence, and by (49), in this case we get the inequality

\[
\|y^{k+1} - y^k\|_{D^{-1}} = \|(I - B)w^k\|_{D^{-1}} \\
\leq \|w^k\|_{D^{-1}} \\
\leq \frac{1 - \delta}{1 - \varepsilon}\|y^k - y^{k-1}\|_{D^{-1}}. \tag{63}
\]

Taking \( \varepsilon := \delta/(1 + \sigma_{J_k}) \), we can summarize the inequalities (61) and (63) as

\[
\|y^{k+1} - y^k\|_{D^{-1}} \leq \left(1 - \frac{1}{2} \frac{\delta \sigma_{J_k}}{1 + \sigma_{J_k}}\right)\|y^k - y^{k-1}\|_{D^{-1}} \\
\leq q \|y^k - y^{k-1}\|_{D^{-1}}, \tag{64}
\]

where

\[
q = \max\left\{\left(1 - \frac{1}{2} \frac{\delta \sigma_J}{1 + \sigma_J}\right) \mid J \in \mathcal{J}\right\}. \tag{65}
\]

This implies the convergence of \( \{y^k\}_{k \geq 0} \) to a vector \( y^* \) and the second estimate in (47). Since \( \|x^{k+1} - x^k\|_{D^{-1}} \leq \|y^{k+1} - y^k\|_{D^{-1}} \), the convergence of \( \{x^k\}_{k \geq 0} \) to a vector \( x^* \) with nonnegative components and the first estimate in (47) follow from (64). The fact that \( x^* \) is a solution of problem (2) with \( \Omega = R^n_+ \) can be deduced directly, as a special case, from Theorem 6. \( \blacksquare \)
Remark 11 The number $\delta$ of (50) plays an important role in the last theorem. Therefore, it is worthwhile to estimate it. Since the equation $Ax = 0$ has no nontrivial solutions with nonnegative components, it follows that there exists a positive constant $a$ such that

$$\langle Bx, x \rangle_{D^{-1}} \geq a \|x\|_{D^{-1}}^2, \text{ for every } x \geq 0.$$  

We claim that

$$\delta \geq \frac{a}{2\rho(B)}. \quad (66)$$

To see this, represent $x = x^0 + x^1$ where $x^0 \in \mathcal{N}(A)$ and $x^1$ is $D^{-1}$-orthogonal to $\mathcal{N}(A)$. Since

$$\langle Bx, x \rangle_{D^{-1}} = \langle Bx^1, x^1 \rangle_{D^{-1}} \leq \rho(B) \|x^1\|_{D^{-1}}^2$$

we have $a \|x\|_{D^{-1}}^2 \leq \rho(B) \|x^1\|_{D^{-1}}^2$, and therefore

$$\|x^0\|_{D^{-1}}^2 \leq \left(1 - \frac{a}{\rho(B)}\right) \|x\|_{D^{-1}}^2$$

from which (66) follows.

In order to verify the inequality $\rho(B) < 2$, which is assumed in Theorem 10, we will need later the following auxiliary lemma.

Lemma 12 Let $G$ be an $n \times n$ real nonnegative matrix. Assume there exist a vector $v > 0$ and a scalar $\kappa > 0$ such that $Gv \leq \kappa v$. Then the spectral radius $\rho(G) \leq \kappa$.

**Proof.** We have that $\rho(G) \leq \|G\|$ for any consistent matrix norm, see, e.g., [Björck, 1996]. Therefore, it is sufficient to find a vector norm such that $\|Gu\| \leq \kappa \|u\|$ for all $u \in \mathbb{R}^n$. Given any $u \in \mathbb{R}^n$ there exist at least one $\lambda > 0$ such that $\|u\| \leq \lambda v$, where $\| \cdot \|$ means component-wise absolute value. Choosing a weighted infinity norm with respect to a vector $v \in \mathbb{R}^n$, defined as

$$\|u\|_{\infty, v} := \min\{\lambda \mid \|u\| \leq \lambda v\},$$  

(68)

gives

$$\|Gu\| \leq \|G\|\|u\| = G\|u\| \leq \|G\|\|u\|_{\infty, v} = \|u\|_{\infty, v}Gv \leq \|u\|_{\infty, v}\kappa v.$$

(69)
Consequently, \[ \|Gu\|_{\infty,v} \leq \kappa \|u\|_{\infty,v}, \] and the proof follows.

To continue the comparison of our work with the description in [Piana and Bertero, 1997], that we started at the end of the previous section, observe that for the nonnegative orthant and a diagonal positive \( D \), (4) implies that the oblique and orthogonal projections coincide. Therefore, we are left with (our) \( D \) matrix inside the iteration formula having a net effect of a component-wise weighting matrix. This option is not present in the adaptation of [Piana and Bertero, 1997] to this case. Using (45), our iteration formula will now be
\[
x^{k+1} = \max(x^k - DA^T W(Ax^k - b), 0),
\]
and will resemble that of [Piana and Bertero, 1997] for this case only if we again put \( D = I \).

4 Supervised Learning

As described in the introduction, some computer vision problems give rise to function approximation problems which can be solved by constructing a model of the approximated function and detecting the correct values of the model’s parameters by a supervised learning process. Such a supervised learning process exposes the model to a number of known functions according to which the model adjusts (“learns”) its parameters. These known functions constitute the training set for the supervised learning process and when this learning process is complete the model is confronted with the practical function approximation problems.

The resulting function approximation problems are quite complex and typically computationally demanding. A tool for handling function approximation problems, within this approach, is to use the concept of channel representation. This helps us to represent a vector (namely, a point) \( z \in R^M \) by some other vector such that there will be an advantage, for the application at hand where \( z \) plays a role, in the new representation. The channel representation, discussed here, performs a mapping of data into a higher-dimensional space. This seems, on the face of it, to be undesirable but it does enable us to introduce locality in the representation, while retaining excellent interpolation properties. We illustrate the basic ideas with a model for the approximation of a function \( f(z) \) and with a learning process of the model’s parameters. Then we use the channel representation to transform \( z \) and \( f(z) \) nonlinearly into higher-dimensional spaces. This allows nonlinear operations to be implemented as locally linear mappings.
Figure 1: The kernel function $\Psi(z)$ in (71).

A mathematical justification for this approach may be traced back to the early paper [Cover, 1965], where the author argues, using probability theory, that a nonlinear pattern-classification problem cast in a high-dimensional space is more likely to be linearly separable there than in a low-dimensional space.

First we describe the notion of channel representation. Our purpose is to represent a vector (namely, a point) $z \in \mathbb{R}^M$ by some other vector such that there will be some advantage, for the application at hand where $z$ plays a role, in the new representation. The precise meaning of this will become clear below. For this purpose we let $\Psi : \mathbb{R}^M \rightarrow \mathbb{R}^H$ denote a kernel function, where the subscript $\geq$ denotes the nonnegative orthant. There exist several different definitions of kernel functions in the literature, but for our purposes we need this function to be smooth, have a bell-shaped maximum and a finite local support. A typical one-dimensional example (for $M = 1$) is

$$
\Psi(z) = \begin{cases} 
\cos^2(\omega(z - c)), & \text{if } \omega |z - c| \leq \pi/2, \\
0, & \text{otherwise,}
\end{cases}
$$

(71)

see Figure 1 for an illustration. Other localized functions such as truncated Gaussians or splines can also be used as kernel functions. We refer to $\omega$ as the kernel width, and to $c$ as the kernel center. We define the nonnegative vector (or point) $a : \mathbb{R}^M \rightarrow \mathbb{R}^H$ by

$$
a(z) = (\Psi_1(z), \Psi_2(z), \ldots, \Psi_H(z))^T,
$$

(72)

where the kernel functions $\Psi_h(z)$ have varying centers, $c_h$, and possibly varying widths, $\omega_h$. We call $a(z)$ a channel representation of $z$ and each component $(a(z))_h$ in the vector is called a channel. Each component, given by the value of its kernel function, is locally-tuned to give high values near the kernel center $c_h$. 

19
Figure 2: An example of the channel representation \( a(z) \) in (72) using kernel functions in (71). (i) The kernel functions \( \Psi_h(z) \), where \( h = 1, 2, \ldots, 10 \) \( (H = 10) \), \( c_h = h \), and \( \omega_h = 3 \), i.e., regularly placed kernels with equal width. (ii) Example of the channel representation \( a(z) \) for \( z = 3.2 \). We get \( a(3.2) = (0, 0.48, 4.78, 2.24, 0, 0, 0, 0, 0, 0)^T \). The figure shows the kernel functions multiplied with their corresponding values for \( z = 3.2 \), i.e., \( \Psi_h(3.2) \Psi_h(z) \). Note that only three elements in \( a(z) \) are nonzero. (iii) Joint channel representation of two values, \( a(3.2) + a(8.5) \).

in \( \mathbb{R}^M \), decreasing values at points further away from \( c_h \), and zero-value outside the support of \( \Psi_h(z) \) (see the one-dimensional example in Figure 1). Figure 2 shows an example of channel representation of a one-dimensional variable. We can interpret the channel representation \( a(z) \) as a position encoding of the value \( z \). The application that we are interested in usually involves a low dimension \( (M) \) for \( z \) and a much higher dimension \( (H) \) for \( a(z) \). The higher memory demands, that stem from the higher dimensions get well-compensated for by the fact that \( a(z) \) is in our application sparse, i.e., only a small portion of its components are nonzero. Furthermore, the position of \( z \) in \( \mathbb{R}^M \) is encoded in the properties of the components of \( a(z) \) with respect to each other. For example, every multiplication of \( a(z) \) by a fixed scalar is a channel representation of the same \( z \). Figure 2(iii) illustrates another powerful property of the channel representation, namely, the ability to represent several values of \( z \) simultaneously, assuming that the values are sufficiently distinct. As we will see later, this property makes the channel representation suitable for dealing with discontinuities in a function.

Now we turn to function approximation. A widely used model assumes that the function \( f : \mathbb{R}^M \to \mathbb{R} \) that needs to be approximated has the form

\[
f(z) = \langle x, a(z) \rangle.
\]  

(73)

In this model \( a(z) \) is a channel representation of \( z \) and the components \( x_h \) of the vector \( x \) are the model parameters. This model is used in Radial Basis Function
Figure 3: Left: Example of kernel functions $\Psi_h(z)$ and $\Psi_n(f(z))$. Right: Nonzero channel values for a single sample $(z, f(z))$. Each local channel $u_n$ is modeled as a linear combination of a few channels $a_h$.

(const of RBF) networks and in Support Vector Machine (SVM) regression in the field of neural networks, see, e.g., [Haykin, 1999; Müller et al., 2001; Moody and Darken, 1989]. Our generalization of this model requires to define $u : \mathbb{R} \to \mathbb{R}^N$ as

$$u(f(z)) = (\Psi_1(f(z)), \Psi_2(f(z)), \ldots, \Psi_N(f(z)))^T,$$

(74)

where the kernel functions $\Psi_n$ need not be identical with those in (72) (we use the same notations for simplicity). The vector $u(f(z))$ is a channel representation of the function value $f(z)$. The proposed model for the function approximation is then

$$u(f(z)) = Xa(z),$$

(75)

where $a(z)$ is, as before, a channel representation of $z$ and $X$, referred to as the linkage matrix, is a matrix whose elements are the model parameters. This model is a linear mapping from a channel representation of $z$ to a channel representation of $f(z)$. Figure 3 illustrates the idea. The model is somewhat similar to fuzzy systems, see, e.g., Kosko [1994], although fuzzy systems are used in low-dimensional problems only.

The function value $f(z)$ can be computed from $u(f(z))$ using a suitable decoding algorithm. Such an algorithm can be found in [Forssén, 2001] for the functions in (71), provided that their kernel centers are at $c_n = n$, for all $n = 1, 2, \ldots, N$, and that they have equal widths $\omega_n = \omega = \pi/K$, for some integer $K \geq 3$. 

21
We reproduce the basic steps of the algorithm here, and refer to [Forssén, 2001] for a more thorough analysis. Denoting by $u_k$ the $k$-th component of the vector $u$, a hypothesis is computed for each $K$ neighboring channels, i.e., for each $n = 1, 2, \ldots, N - K + 1$, a hypothesis is computed as

$$f_n = n + \frac{1}{2\omega} \arg \left( \sum_{k=n}^{n+K-1} u_k e^{i2\omega(k-n)} \right),$$

along with a relevance factor

$$r_n = \begin{cases} \frac{2}{K} \sum_{k=n}^{n+K-1} u_k, & \text{if } n + K/2 - 1 \leq f_n \leq n + K/2, \\ 0, & \text{otherwise}. \end{cases}$$

The function value $f$ is chosen as the hypothesis $f_n$ which has the highest relevance $r_n$. The decoding algorithm is easily modified for the case of non-integer distances, $c_n - c_{n-1} = \Delta$, and $\omega = \pi \Delta / K$, see [Forssén, 2001]. We will denote the decoding operation as $f = \text{dec}(u)$. We emphasize that the decoding should be performed locally in the vector $u$, since a global decoding would ruin the ability to handle several values simultaneously as in Figure 2(iii). This advantage will become clear in the experiment reported below.

In order to apply the model to actual approximation problems, the linkage matrix $X$ needs to be estimated first. This is done by supervised learning using a set of training samples $\{(z_l^l, f(z_l^l))\}_{l=1}^{L}$. Each training sample is a pair, consisting of a given vector and the correct value of the function $f$ for this vector, and it yields another vector pair $(a_l, u_l)$, where $a_l = a(z_l^l)$ and $u_l = u(f(z_l^l))$, again see Figure 3 for an illustration. Let $A$ denote the $H \times L$ matrix having the vector $(a_l^T)$ in its $l$-th column and let $U$ denote the $N \times L$ matrix having the vector $(u_l^T)$ in its $l$-th column. The goal is to find a linkage matrix $X$ such that $u_l^T = X a_l^T$ for all $l$, i.e.,

$$U = X A. \quad (78)$$

This matrix equation usually lacks a solution in practical applications, due to noise and imperfections of the modeling. A common approach is to compute a least-squares solution of (78). Note that $A$ and $U$ are sparse matrices, and that all their elements are nonnegative. It is important, for computational complexity reasons, that the linkage matrix $X$ also be sparse. One way to achieve this is to
constrain the linkage matrix to be nonnegative as well. We will sometimes refer to
the nonnegative constraint as a *monopolar constraint*. Hence, the linkage matrix
$X$ is found as a solution to the constrained least-squares problem
\[
\min \{ e(X) \mid X \geq 0 \}, \tag{79}
\]
where
\[
e(X) = \frac{1}{2} \| XA - U \|_F^2, \tag{80}
\]
and $\| \cdot \|_F$ denotes the Frobenius norm. We can decompose this problem into a set
of smaller, independent, problems. Let $x^n$ denote the $n$-th row vector in $X$, and
let $b^n$ denote the $n$-th row vector in $U$. Thus $b^n$ contains all training data for one
channel $u_n$, and $x^n$ contains all links from the vector $a$ to the channel $u_n$. It is
easy to show that problem (79) is equivalent to optimizing each $x^n$ independently,
i.e.,
\[
\min \{ e_n(x^n) \mid x^n \geq 0 \}, \ n = 1, 2, \ldots, N, \tag{81}
\]
where
\[
e_n(x^n) = \frac{1}{2} \| A^T x^n - b^n \|_2^2. \tag{82}
\]
Sometimes it is useful to have a weighted least-squares problem, where the weight
matrix $W$ is used to weigh each training sample according to its relative impor-
tance. For the sake of simplicity we ignore the weight matrix in the experiment
reported below. The supervised learning problems (81) with the objective func-
tions of (82) are equivalent to the general problem (2) with $\Omega = R^H$ and $A = A^T$.
We will see in the experiment reported below that model (75) after training gives
vectors $u$ similar to the one in Figure 2(iii) near function discontinuities. This
means that the model suggests two hypotheses of $f(z)$ near discontinuities, which
is quite natural. This is not a problem since the decoding is local. Model (73), on
the other hand, will not be able to handle discontinuities correctly, but will give
a value that depends on both hypotheses.

There are several ways to solve the sparse constrained optimization problem
(79). Some of the available methods are documented in [Björck, 1996] and [Adlers,
2000]. However, the high dimensionality largely reduces the number of methods
that can efficiently be applied. The solution to problem (79) is computed in
[Granlund et al., 2002] from a sequence generated by Algorithm 9, with a certain choice of the diagonal matrix $D$. For example, the following choices are used there (although they appear quite differently):

\begin{align*}
D &= \text{diag}^{-1}(AWA^T1), \\
D &= \text{diag}(AW1)\text{diag}^{-1}(AWA^TAW),
\end{align*}

where $1 = (1, 1, \ldots, 1)^T$. These choices can be summarized and generalized by

\begin{equation}
D = \text{diag}(v)\text{diag}^{-1}(AWA^Tv), \text{ for some } v > 0.
\end{equation}

(We include the weight matrix $W$ here temporarily, just to show the general formula.) Note that if $v$ is an eigenvector with eigenvalue $\lambda$ of the matrix $AWA^T$, then we get from Algorithm 9 the ordinary gradient search method for $D = (1/\lambda)I$.

It is well-known that ordinary gradient search without constraints converges for $D = I$ if $0 < \alpha < 2/\rho(AWA^T)$, where $\rho(AWA^T)$ is the largest eigenvalue of $AWA^T$. An advantage in using (84), in shown below the method converges, is that we do not have to compute the largest eigenvalue, which can be a computationally difficult task for large matrices.

We now concentrate on verifying the convergence of Algorithm 9. We let $B = DAWA^T$, where $D$ is chosen according to (84). Then $A \succeq 0$, $W \succeq 0$, and $v > 0$ imply that $B \succeq 0$. Also, we have that

\begin{equation}
Bv = DAWA^Tv = \text{diag}(v)\text{diag}^{-1}(AWA^Tv)AWA^Tv = \text{diag}(v)1 = v.
\end{equation}

Hence, the conditions in Lemma 12 are fulfilled with $\kappa = 1$, and we are guaranteed that $\rho(B) \leq 1$, in fact, we must have $\rho(B) = 1$ since $B$ has an eigenvalue 1. Finally, Theorem 10 assures that any sequence, generated by Algorithm 9, converges to a solution of problem (81)–(82). Note that the nonnegativity of $A$ paved the way for the nonnegativity of $B$ and $D$. For completeness, we notice that the iterative step of Algorithm 9 applied to all problems (81)–(82) simultaneously can be summarized as

\begin{equation}
X(n + 1) = \max(0, X(n) - (X(n)A - U)A^TD),
\end{equation}

which is an iterative solution to problem (79)–(80).

$D$ is sometimes called a preconditioner and can, if properly chosen, allow a considerable acceleration of the ordinary gradient search, see, e.g., [Eicke, 1992] and [Piana and Bertero, 1997]. It is usually a good idea to choose $D$ such that it,
in some way, mimics the behavior of the inverse of $AWA^T$. From (85) we see that $D$ behaves as a (local) inverse for vectors parallel to $v$. We can, therefore, expect a fast convergence if the solution, i.e., each row vector $x^n$ in $X$, is parallel to $v$. Note that we are at least guaranteed that the solution lies in the same orthant as $v$, since they are both nonnegative.

Sometimes we wish to use other constraints such as box constraints. Then a more general algorithm which can solve the least-squares problem with general closed convex set constraints is needed. Algorithm 1 and Theorem 6 provide the necessary theoretical background for such extensions.

5 Experimental Demonstration: Function Approximation

We demonstrate the ideas presented in the previous section with a numerical experiment on learning a function $f : \mathbb{R} \rightarrow \mathbb{R}$. The basic steps are: (i) choose a set of points that will be used as training samples. Second, (ii) compute the channel representation of each training point and if we use model (75) we also need to compute the channel representation of the corresponding function values, and (iii) compute the model parameters by solving a constrained least-squares problem where the error function is based on the training samples.

As our example we choose the function

$$f(z) = \begin{cases} 
  z, & \text{if } z < 0.5, \\
  0.25 - 4(z - 0.75)^2, & \text{if } z \geq 0.5,
\end{cases} \quad (87)$$

see Figure 4. We choose a simple function which makes it easier to interpret the results of the experiment. Nevertheless, it contains a discontinuity which makes it complex enough to show the benefit of the new model (75), i.e., mapping to
the channel representation of \( f(z) \), compared to the old model (73) of mapping to the function \( f(z) \) directly. The experiment is intended to demonstrate a number of things. We compare the convergence properties of algorithm 9 with those of ordinary projected gradient. We also show the sparsity property of the linkage matrix \( X \), due to the nonnegativity constraint. Finally, we show the advantage of the new model (75) compared to the old model (73).

5.1 Experimental Setup

Regardless of which model we choose, we need a set of training samples to compute the model parameters. We choose the points \( z_l = 0, 0.02, 0.04, \ldots, 1 \) \( (L = 51) \), together with the function value in each point, \( f(z_l) \). For each sample \((z_l, f(z_l))\) we then compute the channel representations \( a(z_l) \) and \( u(f(z_l)) \). We will explore two choices of channel representation \( a(z) \) for \( z \):

(I) Regularly placed kernel centers, with equal width. In this case we choose the kernel centers as \( c_h = (h - 1)/19, h = 1, 2, \ldots, 20 \) \( (H = 20) \), and the kernel widths as \( \omega = 19\pi/2 \).

(II) Irregularly placed kernel centers, with varying width. Here we choose \( H = 100 \) kernels with kernel centers randomly placed between 0 and 1 (and sorted afterwards), and with kernel widths as random number between \( \pi/0.15 \) and \( \pi/0.05 \).

See Figure 5 for an illustration. The kernel choice (I) may be a more intuitive choice for this example, but the choice (II) is in our experience more representative for the type of data encountered in many computer vision applications. The matrix \( A \) for the two choices, which contains the channel vectors \( a(z_l) \) for the training samples, will be denoted \( A_I \) and \( A_{II} \) respectively, and are shown in Figure 6. The same figure also shows the singular values for both choices. The Euclidean condition numbers are \( \text{cond}(A_I) \approx 1.4 \) and \( \text{cond}(A_{II}) \approx 3500 \).

For model (75) we also need to compute the channel representation of \( f(z) \). We choose \( N = 7 \) kernels, with regularly placed kernel centers \( c_n = (n - 2)/(N - 3), n = 1, 2, \ldots, 7 \), and equal kernel widths \( \omega = (N - 3)\pi/3 \). The kernels are shown in Figure 5, and Figure 6 shows the the matrix \( U \) which contains the channel vectors \( u(f(z_l)) \) for the training samples.

It is always a good idea to evaluate the performance of a model on data other than that used for training. Therefore we use twice as many points for evaluation as we use for training, i.e., \( z = 0, 0.01, 0.02, \ldots, 1 \). These points are channel
Figure 5: Two choices of channel representation $a(z)$. (I) Regularly placed kernel centers, with equal width. (II) Irregularly placed kernel centers, with varying width. The training samples $\{(z^l, f(z^l))\}$ are also plotted.

Figure 6: Top: $\mathcal{U}$, $\mathcal{A}_I$, and $\mathcal{A}_{II}$ ($\mathcal{U}$ is the same for both experiments (I) and (II)). White denotes value zero and black denotes value one. Bottom: Singular values for $\mathcal{A}_I$ and $\mathcal{A}_{II}$. 

27
encoded, and the resulting channel vectors $a(z)$ are collected in a matrix, which for the two choices of kernels will be denoted by $A^e_1$ and $A^e_2$, respectively (not shown here).

5.2 Models and Methods

We compare three methods to solve the function approximation problem. The first two methods consider the model (75), and we compare two algorithms to solve for the linkage matrix $X$ in problem (79):

Method 1:

Model $u(f(z)) = Xa(z)$, where the linkage matrix $X$ is estimated from ordinary projected gradient, i.e.,

$$X(n + 1) = \max(0, X(n) - \alpha(X(n)A - U)A^T),$$

(88)

where $\alpha = 1/\rho(AA^T)$.

Method 2:

Model $u(f(z)) = Xa(z)$, where the linkage matrix $X$ is estimated from our Algorithm 9, i.e.,

$$X(n + 1) = \max(0, X(n) - (X(n)A - U)A^TD),$$

(89)

with $D$ chosen according to (84), where $v = 1$. Actually, as mentioned before, this is algorithm 9 applied to all problems (81) simultaneously.

The initial value of $X$ is zero in both Method 1 and Method 2. The computational progress of the methods will be measured using the error function

$$err(X) = \frac{\|XA - U\|_F}{\|U\|_F}.$$  

(90)

Note that the progress of (90) displays the convergence of the function model $XA$, which is of more importance in this application than the convergence of the model parameters $X$. It is not yet clear whether the preconditioner in Method 2 also accelerates the convergence of $X$. 

28
We will also compare model (75) with a model that maps to the function values directly, as in (73). In this case we let \( b \) denote the vector containing all function values for the training samples, i.e., \( b = (f(z^1), f(z^2), \ldots, f(z^L))^T \). But to get a fair comparison, we explore a slight generalization of model (73). This is the normalized model,

\[
  f(z) = \frac{\langle x, a \rangle}{\|a(z)\|_p},
\]

where \( \| \cdot \|_p \) denotes the \( p \)-norm. Furthermore, the parameter vector \( x \) is often estimated from a least-squares problem with Tikhonov regularization, i.e.,

\[
  \min_x \left( \frac{1}{2} \|A^T x - b\|^2 + \gamma \|x\|^2 \right).
\]

We, therefore, explored this option too, in comparison with the nonnegative constraint. It was found (not presented here) that model (91) with normalization \( \|a\|_1 = \langle 1, a \rangle \), in combination with Tikhonov regularization, gave the best result. As the third method we, therefore, choose

**Method 3:**

Model \( f = \langle x, a \rangle / \langle 1, a \rangle \), with the estimated parameters

\[
  x = (\hat{A}A^T + \gamma I)^{-1} \hat{A} b,
\]

where \( \hat{A} = A \text{diag}^{-1}(1^T A) \) and \( \gamma \) is chosen by exhaustive search.

It should be noted that the nonnegativity constraint performed fairly well here too, and would have the advantage that it is parameter free.

### 5.3 Computational Results

We now analyze the results of the experiment. We divide the discussion into the two choices of channel representation \( a(z) \) described in Section 5.1.
Experiment (I) - regularly placed channels \( a(z) \): Figure 7 shows the results of experiment (I). We see in Figure 7(a) that both Methods 1 and 2 have approximately the same rate of convergence. This is because \( D \) approximates \( \alpha I \) for well-conditioned matrices \( A \). Both Methods 1 and 2 also produce approximately the same linkage matrix \( X \), see Figure 7(b). The linkage vector \( x \) for Method 3 is shown in Figure 7(c). \( X \) has a sparsity of 42%, which means that \( X \) has about three times more nonzero coefficients than \( x \).

Note that the nonnegativity constraint makes the resulting model more easy to interpret. The channel representation, combined with the linkage matrix \( X \), is implementing a “fuzzy” look-up table. This is so because the channel representations \( a(z) \) and \( u(f(z)) \) can be interpreted as position encodings of the values \( z \) and \( f(z) \), respectively, and because \( X \) is sparse and nonnegative. To see this even more clearly we study the steps involved in estimation of a function value \( f(z) \) given a certain value \( z \) (keep Figures 3 and 7(b) in mind): First we compute the channel representation \( a(z) \). In other words, the value \( z \) “activates” a few channels \( (a(z))_h \). Secondly, we estimate the channel representation of \( f(z) \) from the model, i.e., \( u(f(z)) = Xa(z) \). This means that the position of the value \( z \) is “activating” a position of the function value \( f(z) \), through the linkage matrix \( X \). To be more specific, the fuzzy position of \( z \) which is encoded in channel \( (a(z))_h \) is activating the fuzzy position of \( f(z) \) which is encoded in the channel \( (u(f(z)))_n \) to a certain degree that is determined by the linkage element \( (X)_{nh} \). Finally, we decode the channel vector \( u(f(z)) \) to get the function value \( f(z) \).

Figure 7(d,e) shows the results when we apply model (75) to the evaluation data. Note that the channel vectors \( u(f(z)) \) represents two values near the discontinuity. The discontinuity is preserved when we decode \( u(f(z)) \) to get the function value \( f(z) \), because the decoding algorithm is performed locally in the channel vector as described in the previous section. The result of applying model (73) to the evaluation data is shown in Figure 7(f), and we see that the performance is much worse near the discontinuity.

Experiment (II) - irregularly placed channels \( a(z) \): Figure 8 shows the results of experiment (II). In this case it is seen that Method 2 converges initially faster than Method 1. The use of the preconditioner \( D \) is accelerating the convergence. \( X \) has a sparsity of 39%, which is about 2.7 times as many nonzero coefficients as in \( x \). As seen from Figure 8(e,f), the model (75) again outperforms the model (73) close to the discontinuity. We can also observe a ringing effect around the discontinuity, which is typical of all linear methods.
Figure 7: Results from experiment (I).
(a) Convergence rates for Method 1 (dotted) and Method 2 (solid). The plot shows the error function, $err(X(n))$, in (90) along during iteration.
(b) Resulting linkage matrix $X$ for model $u = Xa$ (approximately the same for both Methods 1 and 2).
(c) Resulting linkage vector $x$ for model $f = x^T a$ using method 3.
(d) Channel representation, $U_i^c = XA_i^c$ (7 × 101).
(e) $\hat{f} = \text{dec}(XA_i^c)$ on evaluation data.
(f) $\hat{f} = x^TA_i^c$, on evaluation data.
Figure 8: Results from experiment (II).
(a) Convergence rates for Method 1 (dotted) and Method 2 (solid). The plot shows the error function $err(X(n))$ in (90) during iteration.
(b) Resulting linkage matrix $X$ for model $u = Xa$ (approximately the same for both Methods 1 and 2).
(c) Resulting linkage vector $x$ for model $f = x^T a$ using method 3.
(d) Channel representation, $U_{II}^c = XA_{II}^c$ (7 × 101).
(e) Resulting function approximation, $\hat{f} = \text{dec}(XA_{II}^c)$, on evaluation data.
(f) Resulting function approximation, $\hat{f} = x^T A_{II}^c$, on evaluation data.
5.4 Monopolar regularization

We argued in Section 3.1, that the monopolar constraint acts as a form of regularization. The upper limit $x^o$ derived in Theorem 7 is quite conservative though. The solution $x^*$ (or $X^*$) is often much lower than $x^o$ (or $X^o$), especially if there are large overlaps between the kernel functions $\Psi_h$. From Theorem 7 it immediately follows that $\|x^\dagger\| \leq \|x^o\|$. We compare the norms of the solutions from the experiments (I) and (II) with the unconstrained solution which is computed as $X = \mathbf{U} \mathbf{A}^T (\mathbf{A} \mathbf{A}^T)^{-1}$. In experiment (I) we get the constrained solution $\|X_1^\dagger\| = 4.83$ and the upper limit $\|X_1^o\| = 6.10$, while the unconstrained solution has $\|X_1\| = 4.83$. The constrained solution and the unconstrained solution is approximately equal in this case. But in experiment (II) we get $\|X_1^{II}\| = 2.84$ and $\|X_1^{II}\| = 13.68$, while the unconstrained solution has $\|X_{II}\| = 325$. We see that the monopolar constraint works as a regularization in the ill-conditioned case (II).

5.5 Conclusions and Discussion

We showed that the choice of preconditioner $D$ in (84) gives an accelerated convergence, compared to ordinary projected gradient search. We also showed that the nonnegative constraint has a regularizing behavior in the type of applications discussed here, and that, in addition, it gives a sparse solution. These statements are supported by numerical experiments.

Computer vision problems often involve high-dimensional functions $f(z) : \mathbb{R}^M \rightarrow \mathbb{R}$. But the fundamental idea is similar to the one described above, namely, to decompose the space and represent the problem by locally linear problems. The kernel functions $\Psi_h : \mathbb{R}^M \rightarrow \mathbb{R}$ are usually not of the simple kind described in (71), but of a very complex nature. They do not have to be local in the domain of $f$, only in the range of $f$, and they are not regularly positioned. An important application is the problem of object pose estimation, where the input to the system is a discretized image of an object and the desired output is the object pose angle. One general attempt to solve this problem is to show a system a set of examples, i.e., images of objects and their corresponding poses, from which the system learns an approximation of the true mapping. To be more specific, one idea that has been a focus of research is to detect local interest points on the object such as corners, and then compute feature vectors $z$ based on the image content in a local region around each point. One advantage of using local information is to become more robust to occlusions. We can also compute feature vectors from homogeneous region descriptors, local orientation, or even more complex features.
such as a combination of several corner points. These feature vectors are used as input to a function approximation problem, where the output is the desired pose. To use the method presented in this report we need a channel representation of the feature vector $z$ and the pose angle. The channel vector $a(z)$ typically has a dimensionality in the order of magnitude of $10^4 - 10^5$, and we may have the same order of magnitude of training samples. This function approximation problem is often of a very complex nature and few practical solutions exists today. The choice of reliable feature vectors is also a difficult problem in itself and a topic for research. We therefore evaluated the algorithm presented here using the more limited problem of estimating a function $f: \mathbb{R} \rightarrow \mathbb{R}$. But the algorithm has been successfully used in object pose estimation experiments as described above, see, e.g., [Granlund and Moe, 2002], and we feel that this approach, i.e., to decompose the problem into locally linear problems, will be a useful tool also in various other computer vision problems.

It is important to be able to handle discontinuous functions in many applications, e.g., an adaptive filtering algorithm, used to remove noise in images should not smooth sharp edges. It is also important to be able to handle several values simultaneously, one application being object recognition. Natural images usually contain more than one object, and a computer vision system should be able to handle several objects (i.e., several function values $f(z)$) simultaneously from one single image. These problems can be handled using a channel representation $u(f(z))$. As mentioned before, this representation can handle several values simultaneously without interference, assuming that the values are sufficiently well-separated. In some application we may not even want to decode the channel vector $u$, but instead pass it on directly to higher level processes. Model (73) on the other hand cannot handle several values, and tends to give ringing effects around discontinuities.

**Acknowledgments**

We thank Professor Lars Eldén for calling our attention to the paper [Piana and Bereroto, 1997]. This work was partially supported by the Swedish Research Council through a grant for the project *A New Structure for Signal Processing and Learning*, and by the Swedish Foundation for Strategic Research through the project *VISIT (VIsual Information Technology)*. The work of Y. Censor was done in part at the Center for Computational Mathematics and Scientific Computation (CCMSC) of the University of Haifa and was supported by grant 592/00 34
of the Israel Science Foundation, founded by the Israel Academy of Sciences and Humanities.

References


