Efficient Dictionary Learning with Sparseness-Enforcing Projections

Markus Thom · Matthias Rapp · Günther Palm

Received: 31 January 2014 / Accepted: 7 January 2015 / Published online: 11 February 2015

Abstract Learning dictionaries suitable for sparse coding instead of using engineered bases has proven effective in a variety of image processing tasks. This paper studies the optimization of dictionaries on image data where the representation is enforced to be explicitly sparse with respect to a smooth, normalized sparseness measure. This involves the computation of Euclidean projections onto level sets of the sparseness measure. While previous algorithms for this optimization problem had at least quasi-linear time complexity, here the first algorithm with linear time complexity and constant space complexity is proposed. The key for this is the mathematically rigorous derivation of a characterization of the projection's result based on a soft-shrinkage function. This theory is applied in an original algorithm called Easy Dictionary Learning (EZDL), which learns dictionaries with a simple and fast-to-compute Hebbian-like learning rule. The new algorithm is efficient, expressive and particularly simple to implement. It is demonstrated that despite its simplicity, the proposed learning algorithm is able to generate a rich variety of dictionaries, in particular a topographic organization of atoms or separable atoms. Further, the dictionaries are as expressive as those of benchmark learning algorithms in terms of the reproduction quality on entire images, and result in an equivalent denoising performance. EZDL learns approximately 30 % faster than the already very efficient Online Dictionary Learning algorithm, and is therefore eligible for rapid data set analysis and problems with vast quantities of learning samples.

Keywords Sparse coding · Sparse representations · Dictionary learning · Explicit sparseness constraints · Sparseness-enforcing projections

Communicated by Julien Mairal, Francis Bach, Michael Elad.

M. Thom (✉) · M. Rapp
driveU / Institute of Measurement, Control and Microtechnology, Ulm University, 89081 Ulm, Germany
e-mail: markus.thom@uni-ulm.de

M. Rapp
e-mail: matthias.rapp@uni-ulm.de

G. Palm
Institute of Neural Information Processing, Ulm University, 89081 Ulm, Germany
e-mail: guenther.palm@uni-ulm.de

1 Introduction

In a great variety of classical machine learning problems, sparse solutions are attractive because they provide more efficient representations compared to non-sparse solutions. There is an overwhelming evidence that mammalian brains respect the sparseness principle (Laughlin and Sejnowski 2003), which holds true especially for the mammalian visual cortex (Hubel and Wiesel 1959; Olshausen and Field 1996, 1997). It suggests itself that sparseness be a fundamental prior to a variety of signal processing tasks. In particular, this includes low-level image processing since natural images can be represented succinctly using structural primitives (Olshausen and Field 1997; Mairal et al. 2009b). Interesting and biologically plausible sparse representations were discovered through computer simulations on natural images (Olshausen and Field 1996, 1997). Related representations can be obtained by analysis of temporal image sequences (van Hateren and Ruderman 1998; Olshausen 2003), stereo image pairs and images with chromatic information (Hoyer and Hyvärinen 2000), or by enforcing a topographic organization (Hyvärinen et al. 2001; Kavukcuoglu et al. 2009).

Sparseness alleviates the effects of random noise in a natural way since it prevents arbitrary combinations of measured signals (Donoho 1995; Hyvärinen 1999; Elad 2006). In fact, methods based on sparse representations were shown to achieve state-of-the-art performance for image denoising (Mairal et al. 2009b). Further notable image processing applications that benefit from the efficiency gained through sparseness are as diverse as deblurring (Dong et al. 2011), super-resolution (Yang et al. 2010, 2012; Dong et al. 2011), compression (Skretting and Engan 2011; Horev et al. 2012), and depth estimation (Tošić et al. 2011).

1.1 Dictionary Learning and Sparseness Measures

Each of these tasks needs a model capable of reproducing the signals to be processed. In a *linear generative model for sparse coding,* a sample $x \in \mathbb{R}^d$ with d features should be expressed approximately as a linear combination of only a few atoms of a larger dictionary:

$$x \approx Wh, \text{ where } W \in \mathbb{R}^{d \times n} \text{ and } h \in \mathbb{R}^n \text{ is sparsely populated.}$$

Here, W is the *dictionary* which is fixed for all samples, and h is a *sparse code word* that depends on the concrete sample. The n columns of W represent the *atoms*, which are also called bases or filters. This sparse coding framework is well-suited for overcomplete representations where $n \gg d$. The dictionary can be generated by wavelets, for example, or adapted to a specific task by solving an optimization problem on measurement data. The latter is also called *dictionary learning* in this context.

Sparseness acts as a regularizer. If h was not constrained to sparseness, then trivial choices of W would suffice for perfect reproduction capabilities. But when h is sparse, then x can be represented by additive superposition of only a small number of bases, thus preventing trivial solutions.

A fundamental problem when working with sparse representations is how to decide on a function that formally assesses the sparseness of a vector. The L_0 pseudo-norm

$$\|\cdot\|_0 : \mathbb{R}^n \to \{0, \dots, n\}, \quad x \mapsto |\{i \in \{1, \dots, n\} | x_i \neq 0\}|,$$

simply counts the nonzero entries of its argument. It is a poor choice since it is non-continuous, prone to random noise and fails to fulfill desirable properties of meaningful sparseness measures (Hurley and Rickard 2009).

Throughout this paper, we will use the smooth, normalized sparseness measure σ proposed by Hoyer (2004):

$$\sigma : \mathbb{R}^n \backslash \{0\} \to [0,\ 1], \quad x \mapsto \frac{\sqrt{n} - \|x\|_1 / \|x\|_2}{\sqrt{n} - 1}.$$

Here, $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the Manhattan norm and the Euclidean norm, respectively. The normalization has been designed such that σ attains values between zero and one. When $x \in \mathbb{R}^n$ satisfies $\sigma(x) = 1$, then all entries of x but one vanish. Conversely, when $\sigma(x) = 0$, then all the entries of x are equal. The function σ interpolates smoothly between

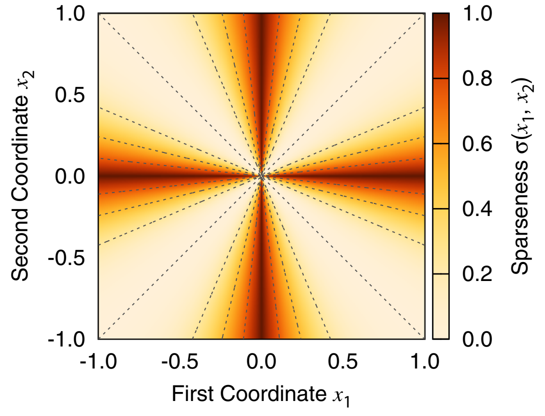


Fig. 1 Visualization of Hoyer's sparseness measure σ. The abscissa and the ordinate specify the entries of a two-dimensional vector, the obtained sparseness degree is color coded. *The dashed lines* are contour levels at intervals of 0.25

these extremes, see Fig. 1. Moreover, it is scale-invariant so that the same sparseness degree is obtained when a vector is multiplied with a nonzero number. Hence if a quantity is given in other units, for example in millivolts instead of volts, no adjustments whatsoever have to be made.

The sparseness degree with respect to σ does not change much if a small amount is added to all entries of a vector, whereas the L_0 pseudo-norm would indicate that the new vector is completely non-sparse. These properties render Hoyer's sparseness measure intuitive, especially for non-experts. It has been employed successfully for dictionary learning (Hoyer 2004; Potluru et al. 2013), and its smoothness results in improved generalization capabilities in classification tasks compared to when the L_0 pseudo-norm is used (Thom and Palm 2013).

1.2 Explicit Sparseness Constraints and Projections

A common approach to dictionary learning is the minimization of the reproduction error between the original samples from a learning set and their approximations provided by a linear generative model under sparseness constraints (Olshausen and Field 1996, 1997; Kreutz-Delgado et al. 2003; Mairal et al. 2009a). It is beneficial for practitioners and end-users to enforce *explicit* sparseness constraints by demanding that all the code words h in a generative model possess a target sparseness degree of $\sigma_H \in (0,\ 1)$. This leads to optimization problems of the form

$$\min_{W,h} \|x - Wh\|_2^2 \text{ so that } \sigma(h) = \sigma_H.$$

Here, the objective function is the reproduction error implemented as Euclidean distance. *Implicit* sparseness constraints, on the other hand, augment the reproduction error with an additive penalty term, yielding optimization problems such as

$$\min_{W,\, h} \ \|x - Wh\|_2^2 + \lambda \, \|h\|_1 \, .$$

Here, $\lambda > 0$ is a trade-off constant and the Manhattan norm is used to penalize non-sparse code words as convex relaxation of the L_0 pseudo-norm (Donoho 2006).

Trading off the reproduction error against an additive sparseness penalty is non-trivial since the actual resulting code word sparseness cannot easily be predicted. Explicit constraints guarantee that the adjusted sparseness degree is met, making tuning of intransparent scale factors such as λ in the example above obsolete. This way, one can concentrate on the actual application of the theory rather than having to develop an intuition of the meaning of each and every parameter.

The mathematical tool to achieve explicit sparseness is a *sparseness-enforcing projection operator*. This is a vector-valued function which maps any given point in Euclidean space to its nearest point that achieves a pre-set target sparseness degree. One use case of this theory is projected gradient methods (Bertsekas 1999), where a given objective function should be optimized subject to hard side conditions. Replacing the parameters with their best approximations lying in a certain set after each update step ensures that the constraints are satisfied during optimization progress.

1.3 Contributions of this Paper and Related Work

This paper studies dictionary learning under explicit sparseness constraints with respect to Hoyer's sparseness measure σ. A major part of this work is devoted to the efficient algorithmic computation of the sparseness-enforcing projection operator, which is an integral part in efficient dictionary learning. Several algorithms were proposed in the past to solve the projection problem (Hoyer 2004; Theis et al. 2005; Potluru et al. 2013; Thom and Palm 2013). Only Thom and Palm (2013) provided a complete and mathematically satisfactory proof of correctness for their algorithm. Moreover, all known algorithms have at least quasi-linear time complexity in the dimensionality of the vector that should be projected.

In this paper, we first derive a characterization of the sparseness projection and demonstrate that its computation is equivalent to finding the root of a real-valued auxiliary function, which constitutes a much simpler problem. This result is used in the proposition of an algorithm for the projection operator that is asymptotically optimal in the sense of complexity theory, that is the time complexity is linear and the space complexity is constant in the problem dimensionality. We show through experiments that when run on a real computing machine the newly proposed algorithm is far superior in its computational demands to previously proposed techniques, even for small problem dimensionalities.

Existing approaches to dictionary learning that feature explicit sparseness constraints can be categorized into ones that use Hoyer's σ and ones that employ the L_0 pseudo-norm. Hoyer (2004) and Potluru et al. (2013) considered matrix factorization frameworks subject to σ constraints, with space requirements linear in the number of learning samples which prevents processing large data sets. Thom and Palm (2013) designed sparse code words as the result of the sparseness-enforcing projection operator applied to the product of the dictionary with the samples. This requires the computation of the projection's gradient during learning, which is feasible yet difficult to implement and has non-negligible adverse effects on the execution time.

The following approaches consider explicit L_0 pseudo-norm constraints: Aharon et al. (2006), Skretting and Engan (2010) and Coates and Ng (2011) infer sparse code words in each iteration compatible to the data and dictionary by employing basis pursuit or matching pursuit algorithms, which has a negative impact on the processing time. Zelnik-Manor et al. (2012) consider block-sparse representations, here the signals are assumed to reside in the union of few subspaces. Duarte-Carvajalino and Sapiro (2009) propose to simultaneously learn the dictionary and the sensing matrix from example image data, which results in improved reconstruction results in compressed sensing scenarios.

In addition to the contributions on sparseness projection computation, this paper proposes the *Easy Dictionary Learning* (EZDL) algorithm. Our technique aims at dictionary learning under explicit sparseness constraints in terms of Hoyer's sparseness measure σ using a simple, fast-to-compute and biologically plausible Hebbian-like learning rule. For each presented learning sample, the sparseness-enforcing projection operator has to be carried out. The ability to perform projections efficiently makes the proposed learning algorithm particularly efficient: 30 % less training time is required in comparison to the optimized Online Dictionary Learning method of Mairal et al. (2009a).

Extensions of Easy Dictionary Learning facilitate alternative representations such as topographic atom organization or atom sparseness, which includes for example separable filters. We furthermore demonstrate the competitiveness of the dictionaries learned with our algorithm with those computed with alternative sophisticated dictionary learning algorithms in terms of reproduction and denoising quality of natural images. Since other tasks, such as deblurring or super-resolution, build upon the same optimization problem in the application phase as reproduction and denoising, it can be expected that EZDL dictionaries will exhibit no performance degradations in those tasks either.

The remainder of this paper is structured as follows. Section 2 derives a linear time and constant space algorithm for the computation of sparseness-enforcing projections. In Sect. 3, the Easy Dictionary Learning algorithm for explicitly sparseness-constrained dictionary learning is proposed. Section 4 reports experimental results on the performance of the newly proposed sparseness projection algorithm and the Easy Dictionary Learning algorithm. Section 5 concludes the paper with a discussion, and the appendix contains technical details of the mathematical statements from Sect. 2.

2 Efficient Sparseness-Enforcing Projections

This section proposes a linear time and constant space algorithm for computation of projections onto level sets of Hoyer's σ. Formally, if $\sigma^* \in (0,\ 1)$ denotes a *target sparseness degree* and $x \in \mathbb{R}^n$ is an arbitrary point, the point from the level set $S := \{s \in \mathbb{R}^n | \sigma(s) = \sigma^*\}$ that minimizes the Euclidean distance to x is sought. The function that computes $\arg\min_{s \in S} \|x - s\|_2$ is also called *sparseness-enforcing projection operator* since the situation where $\sigma(x) < \sigma^*$ is of particular interest.

Due to symmetries of σ, the above described optimization problem can be reduced to finding the projection of a vector $x \in \mathbb{R}^n_{\geq 0}$ with non-negative coordinates onto the set

$$T := \{s \in \mathbb{R}^n_{\geq 0} | \ \|s\|_1 = \lambda_1 \text{ and } \|s\|_2 = \lambda_2\} \subseteq S,$$

where λ_1 and λ_2 are target norms that should be chosen such that $\sigma^* = \left(\sqrt{n} - {}^{\lambda_1}/_{\lambda_2}\right) \big/ \left(\sqrt{n} - 1\right)$ (Hoyer 2004; Thom and Palm 2013). With this choice, σ constantly attains the value of σ^* on the entire set T, hence T is a subset of S.

Thom and Palm (2013) proved that T is the intersection of a scaled canonical simplex and a hypercircle. They further demonstrated that projections onto T can be computed with a finite number of alternating projections onto these two geometric structures. The proof of correctness of this method could not rely on the classical alternating projection method (see for example Deutsch 2001) due to the lacking convexity of T, rendering the proof arguments quite complicated.

The remainder of this section proceeds as follows. First, a succinct characterization of the sparseness projection is given. It is then shown that computing sparseness projections is equivalent to finding the zero of a monotone real-valued function. We prove that this can be achieved with optimal asymptotic complexity by proposing a new algorithm that solves the projection problem.

2.1 Representation of the Projection

The main theoretical result of this paper is a representation theorem that characterizes the projection onto T. This was gained through an analysis of the intermediate points that emerge from the alternating projections onto a simplex and a hypercircle. A closed-form expression can then be provided by showing that the intermediate points in the projection algorithm satisfy a loop-invariant, a certain property fulfilled after each step of alternating projections.

Since a mathematical rigorous treatment of this result is very technical, it is deferred to the appendix. We assume here that the vector x to be projected is given so that its projection is unique. Since it is guaranteed that for all points except for a null set there is exactly one projection (Theis et al. 2005, Theorem 2.6), we can exclude points with non-unique projections in our considerations, which is no restriction in practice. We may now state a characterization of the projection outcome:

Representation Theorem *Let $x \in \mathbb{R}^n_{\geq 0}$ and let $p \in T$ denote the unique projection of x onto T. Then there is exactly one real number α^* with*

$$p = \frac{\lambda_2}{\|q\|_2} \cdot q \text{ where } q := \max\left(x - \alpha^* \cdot e,\ 0\right) \in \mathbb{R}^n.$$

Here, $e \in \{1\}^n$ is the n-dimensional vector where all entries are unity. If the indices of the positive coordinates of p are known, then α^* can be computed directly from x with an explicit expression.

In words, the projection p is the point q rescaled to obtain an L_2 norm of λ_2. The vector q is computed from the input vector x by subtracting the scalar α^* from all the entries, and afterwards setting negative values to zero. It is remarkable that the projection admits such a simple representation although the target set for the projection is non-convex and geometrically quite complicated.

The function that maps a scalar $\xi \in \mathbb{R}$ to $\max(\xi - t,\ 0)$ for a constant offset $t \in \mathbb{R}$ is called the *soft-shrinkage function*. If $t = 0$, it is also called the *rectifier function*. Because the central element of the projection representation is a soft-shrinkage operation applied entry-wise to the input vector, carrying out projections onto level sets of Hoyer's sparseness measure can be interpreted as denoising operation (Donoho 1995; Hyvärinen 1999; Elad 2006).

2.2 Auxiliary Function for the Sparseness Projection

The projection problem can hence be reduced to determining the soft-shrinkage offset α^*, which is a one-dimensional problem on the real line. Further, it is reasonable that α^* must be smaller than the maximum entry of x since otherwise q would be the null vector, which is absurd. In the usual case where $\sigma(x) < \sigma^*$ we can further conclude that α^* must be non-negative. Otherwise, the result of the projection using the representation theorem would be less sparse than the input, which is impossible. Therefore, the projection problem can be further reduced to finding a number from the bounded interval $[0,\ x_{\max})$ with $x_{\max} := \max_{i \in \{1,\ldots,n\}} x_i$.

Next, a method for efficiently deciding whether the correct offset has been found is required. Similar to the projection onto a canonical simplex (Liu and Ye 2009), we can design a real-valued function that vanishes exactly at the wanted

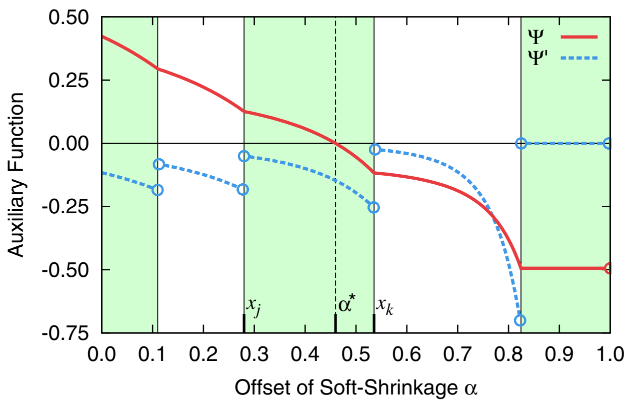


Fig. 2 Plot of the auxiliary function Ψ and its derivative for a random vector x. The derivative Ψ' was scaled using a positive number for improved visibility. The steps in Ψ' are exactly the places where α coincides with an entry of x. It is enough to find an α with $\Psi(x_j) \geq 0$ and $\Psi(x_k) < 0$ for the neighboring entries x_j and x_k in x, because then the exact solution α^* can be computed with a closed-form expression

offset. The properties of Hoyer's σ allow us to formulate this function in an intuitive way. We call

$$\Psi\colon\ [0,\ x_{\max}) \to \mathbb{R},\quad \alpha \mapsto \frac{\|\max(x - \alpha \cdot e,\ 0)\|_1}{\|\max(x - \alpha \cdot e,\ 0)\|_2} - \frac{\lambda_1}{\lambda_2},$$

the *auxiliary function* for the projection onto T.

The rationale for the concrete definition of Ψ is as follows. Due to the representation theorem we know that the projection p onto T is merely a scaled version of the point $q := \max(x - \alpha^* \cdot e,\ 0)$. Moreover $\sigma(p) = \sigma^*$, and due to the scale-invariance of Hoyer's σ follows $\sigma(q) = \sigma^*$. The essence of σ is the ratio of the L_1 norm to the L_2 norm of its argument. Hence here the scale constants that make σ normalized to the interval $[0,\ 1]$ are omitted and the target norms are used instead. We therefore have that $\|q\|_1/\|q\|_2 = \lambda_1/\lambda_2$, and thus $\Psi(\alpha^*) = 0$. As α^* is unique, we can conclude that no other offset of the soft-shrinkage operation leads to a correct solution. In fact, if $\alpha \neq \alpha^*$ and when we write $\tilde{q} := \max(x - \alpha \cdot e,\ 0)$, then we have that $\sigma(\tilde{q}) \neq \sigma^*$ and therefore $\Psi(\alpha) \neq 0$.

An exemplary plot of Ψ is depicted in Fig. 2. Clearly, the auxiliary function is continuous, differentiable except for isolated points and strictly decreasing except for its final part where it is constant. Since the constant part is always negative and starts at the offset equal to the second-largest entry $x_{\text{2nd-max}} := \max\{x_i \mid i \in \{1, \ldots, n\} \text{ and } x_i \neq x_{\max}\}$ of x, the feasible interval can be reduced even more. The step discontinuities in Ψ' coincide exactly with the entries of the input vector x. These appealing analytical properties greatly simplify computation of the zero of Ψ, since standard root-finding algorithms such as bisection or Newton's method (see for example Traub 1964) can be employed to numerically find α^*.

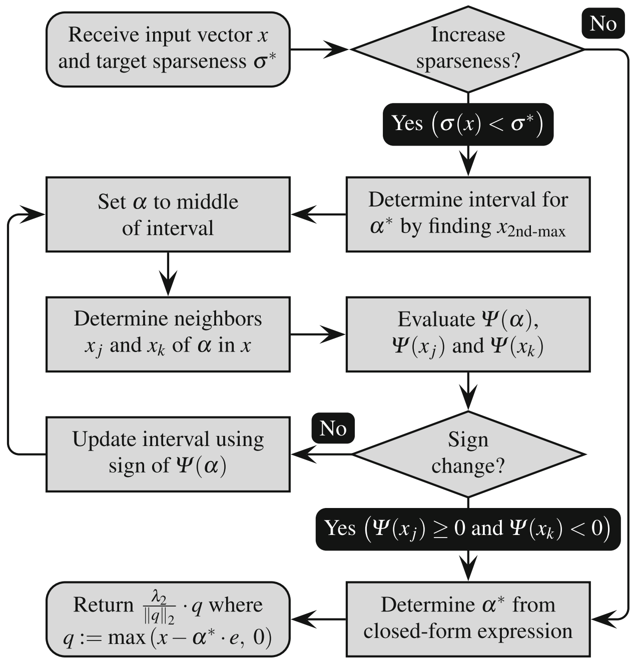

Receive input vector x and target sparseness σ^*
Increase sparseness?
No
Yes $(\sigma(x) < \sigma^*)$
Determine interval for α^* by finding $x_{\text{2nd-max}}$
Set α to middle of interval
Determine neighbors x_j and x_k of α in x
Evaluate $\Psi(\alpha)$, $\Psi(x_j)$ and $\Psi(x_k)$
Sign change?
No
Update interval using sign of $\Psi(\alpha)$
Yes $(\Psi(x_j) \geq 0$ and $\Psi(x_k) < 0)$
Determine α^* from closed-form expression
Return $\frac{\lambda_2}{\|q\|_2} \cdot q$ where $q := \max(x - \alpha^* \cdot e,\ 0)$

Fig. 3 Flowchart of the proposed algorithm for computing sparseness-enforcing projections. The algorithm starts with the box at the *upper left* and terminates with the box at the *lower left*

2.3 Linear Time and Constant Space Projection Algorithm

We can improve on merely a numerical solution by exploiting the special structure of Ψ to yield an analytical solution to the projection problem. This is due to the closed-form expression for α^* which requires the indices of the coordinates in which the result of the projection is positive to be known. The other way around, when α^* is known, these indices are exactly the ones of the coordinates in which x is greater than α^*. When an offset sufficiently close to α^* can be determined numerically, the appropriate index set can be determined and thus the exact value of α^*.

The decision if a candidate offset α is close enough to the true α^* can be made quite efficiently, since α^* is coupled via index sets to individual entries of the input vector x. Hence, it suffices to find the right-most left neighbor x_j of α in the entries of x, and analogously the left-most right neighbor x_k (see Fig. 2 for an example). Whenever $\Psi(x_j) \geq 0$ and $\Psi(x_k) < 0$, the zero α^* of Ψ must be located between x_j and x_k for continuity reasons. But then all values in x greater than α^* are exactly the values greater than or equal to x_k, which can be determined by simply scanning through the entries of x.

Based upon these considerations, a flowchart of our proposed method for computing sparseness-enforcing projections is depicted in Fig. 3. The algorithm performs bisection and continuously checks for sign changes in the auxiliary function Ψ. As soon as this is fulfilled, α^* is computed and the result of the projection is determined in-place. A complete formal presentation and discussion of our proposed algorithm is given in the appendix.

Each intermediate step of the algorithm (that is, determination of the second-largest entry of x, evaluation of the auxiliary function, projection result computation by application of soft-shrinkage and scaling) can be carried out in a time complexity linear in the problem dimensionality n and a space complexity completely independent of n.

Therefore, to show the overall algorithm possesses the same asymptotic complexity, it has to be proved that the number of bisection iterations required for finding α^* is independent of n. The length of the initial interval is upper-bounded by $x_{\text{2nd-max}}$ as discussed earlier. Bisection is moreover guaranteed to stop at the latest once the interval length is smaller than the minimum pairwise distance between distinct entries of x,

$$\delta := \min\{x_k - x_j \mid x_j, x_k \in \mathscr{X} \text{ and } x_j < x_k\} > 0$$

where $\mathscr{X} := \{x_i \mid i \in \{1, \ldots, n\}\}$. This is due to the fact that it is enough to find a sufficiently small range to deduce an analytical solution. The required number of bisection iterations is then less than $\lceil \log_2(x_{\text{2nd-max}}/\delta) \rceil$, see Liu and Ye (2009). This number is bounded from above regardless of the dimensionality of the input vector since $x_{\text{2nd-max}}$ is upper-bounded and δ is lower-bounded due to finite machine precision (Goldberg 1991).

Hence our sparseness-enforcing projection algorithm is asymptotically optimal in the sense of complexity theory. There may still be hidden constants in the asymptotic notation that render the proposed algorithm less efficient than previously known algorithms for a small input dimensionality. Experiments in Sect. 4 demonstrate that this is not the case.

3 Explicitly Sparseness-Constrained Dictionary Learning

This section proposes the *Easy Dictionary Learning* (EZDL) algorithm for dictionary learning under explicit sparseness constraints. First, we introduce an ordinary formulation of the learning algorithm. Sparse code word inference with the sparseness projection algorithm proposed in Sect. 2 renders EZDL efficient and particularly simple to implement. We further discuss extensions that allow concentrating on different aspects of the data set under investigation, such as topographic organization of the atoms and atom sparseness. Only little implementation effort is required for these extensions and their computational demands are low. A description of the comprehensive EZDL learning algorithm is accompanied with several strategies that improve the optimization performance.

3.1 EZDL—Easy Dictionary Learning

Conventional dictionary learning algorithms operate in an alternating fashion, where code words are inferred with a costly optimization procedure before updating the dictionary. EZDL learns a dictionary by first yielding sparse code words through simple inference models and then tuning the dictionary with a simple update step. An *inference model* is a function $I: \mathbb{R}^n \to \mathbb{R}^n$ which accepts *filter responses* in the form of the product of the dictionary $W \in \mathbb{R}^{d \times n}$ with a learning sample $x \in \mathbb{R}^d$, $u := W^T x \in \mathbb{R}^n$, and produces a representation $h := I(u)$ with certain desirable properties. Here, the combination with the sparseness-enforcing projection operator is particularly interesting since it provides a natural method to fulfill explicit sparseness constraints.

We call the choice $I(u) := \Pi_{\sigma_H}(u)$ the *ordinary inference model*, where Π_{σ_H} denotes the Euclidean projection onto the set of all vectors that achieve a sparseness degree of $\sigma_H \in (0,\ 1)$ with respect to Hoyer's sparseness measure σ. This computation can also be interpreted as the trivial first iteration of a projected Landweber procedure for sparse code word inference (Bredies and Lorenz 2008).

The dictionary W is adapted to new data by minimizing the deviation between a learning sample x and its approximation Wh through the linear generative model. The goodness of the approximation is here assessed with a differentiable similarity measure $\rho: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, so that the EZDL optimization problem becomes:

$$\min_W \ \rho(Wh,\ x).$$

Note that h is not a variable of the optimization problem since it is defined as output of an inference model. For the same reason, h does not need to be constrained further as it inherently satisfies explicit sparseness constraints.

Although a wide range of similarity measures is possible, we decided to use the (Pearson product-moment) correlation coefficient since it is invariant to affine-linear transformations (Rodgers and Nicewander 1988). In the context of visual data set analysis, this corresponds to invariance with respect to DC components and gain factors. Moreover, differentiability of this similarity measure facilitates gradient-based learning (Thom and Palm 2013).

If $\rho': \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is the transposed derivative of ρ with respect to its first argument and $g := \rho'(Wh,\ x) \in \mathbb{R}^d$ denotes its value in Wh, the gradient of E_{EZDL} for the dictionary is given by

$$\left(\frac{\partial E_{\text{EZDL}}}{\partial W}\right)^T = gh^T \in \mathbb{R}^{d \times n}.$$

Therefore, when W is tuned with gradient descent a simple and biologically plausible Hebbian-like learning rule (Hyvärinen et al. 2009) results:

$W^{\text{new}} := W - \eta \cdot gh^T$, where $\eta > 0$ denotes the step size.

This update step can be implemented efficiently with simple rank-1 updates (Blackford et al. 2002).

We here assumed $h = I(W^T x)$ to be constant and neglected that it actually depends on W. Without this assumption, the gradient would have to be extended with an additive term which comprises the gradient of the inference model (Thom and Palm 2013, Proposition 39). This, in turn, requires the computation of the sparseness-enforcing projection operator's gradient. This gradient can be computed with simple vector operations as we show in the appendix. However, this constitutes a significant computational burden compared to the simple and fast-to-compute EZDL update step. Section 4 demonstrates through experiments that this simplification is still perfectly capable of learning dictionaries.

3.2 Topographic Atom Organization and Atom Sparseness

Topographic organization of the dictionary's atoms similar to self-organizing maps (Kohonen 1990) or topographic independent component analysis (Hyvärinen et al. 2001) can be achieved in a straightforward way with EZDL using alternative inference models proposed in this section. For this, the dictionary atoms are interpreted to be arranged on a two-dimensional grid. A spatial pooling operator subject to circular boundary conditions can then be used to incorporate interactions between atoms located adjacent on the grid. This can, for example, be realized by averaging each entry of the filter responses in a 3×3 neighborhood on the grid. This convolution operation can be expressed as linear operator applied to the vector $u \in \mathbb{R}^n$, represented by a sparsely populated matrix $G \in \mathbb{R}^{n \times n}$. With G containing all atom interactions, we call $I(u) := \Pi_{\sigma_H}(Gu)$ the *topographic inference model*.

An alternative realization of topographic organization is enforcing *structure sparseness* on the filter responses reshaped as matrix to account for the grid layout. Structure sparseness can be assessed by application of a sparseness measure to the vector with the singular values of a matrix. When the L_0 pseudo-norm is used, then this is the rank of a matrix. The matrix rank can also be measured more robustly through the ratio of the Schatten 1-norm to the Schatten 2-norm, which is essentially Hoyer's sparseness measure σ applied to the singular values (Lopes 2013).

Clearly, any matrix $M \in \mathbb{R}^{a \times b}$ where $a, b \in \mathbb{N}$ can be reshaped to a vector $m := \operatorname{vec}(M) \in \mathbb{R}^{ab}$ with the vectorization operator that stacks all columns of M on top of another (Neudecker 1969). This linear operation can be inverted provided the shape of the original matrix is known, $M = \operatorname{vec}^{-1}_{a \times b}(m)$. In the following, let Π_{κ^*} denote the projection onto the set of all matrices that possess a target rank $\kappa^* \in \{1, \ldots, \min\{a, b\}\}$. A classical result states that any matrix can be projected onto the set of low-rank matrices by computation of its singular value decomposition, applying an L_0 pseudo-norm projection to the vector of singular values thus retaining only the largest ones, and finally computing the reconstruction using the modified singular values (Eckart and Young 1936).

Based on these considerations, let r and c with $n = rc$ be natural numbers describing the topographic grid layout dimensionalities. The *rank-κ_H topographic inference model* is then defined by the composition

$$I(u) := \left(\operatorname{vec} \circ \Pi_{\kappa_H} \circ \operatorname{vec}^{-1}_{r \times c} \circ \Pi_{\sigma_H}\right)(u).$$

In words, this inference model operates as follows. It first approximates the filter responses u with a sparsely populated vector that attains a sparseness of σ_H with respect to Hoyer's σ. These sparsified filter responses are then laid out on a grid and replaced with those best approximating filter responses that meet a lower rank of $\kappa_H \in \{1, \ldots, \min\{r, c\}\}$. Finally, the grid layout is reshaped to a vector since this should be the output type of each inference model.

If $\kappa_H = 1$, then there are two vectors $y \in \mathbb{R}^r$ and $z \in \mathbb{R}^c$ such that $\operatorname{vec}^{-1}_{r \times c}(I(u)) = yz^T$, that is the filter responses can be expressed as a dyadic product when reshaped to a grid. We will demonstrate with experiments that this results in an interesting topographic atom organization similar to independent subspace analysis (Hyvärinen and Hoyer 2000).

Analogous to non-negative matrix factorization with sparseness constraints (Hoyer 2004), the atoms can be enforced to be sparse as well. To achieve this, it is sufficient to apply the sparseness projection to each column of the dictionary after each learning epoch:

$$W^{\text{new}} e_i := \Pi_{\sigma_W}(W e_i) \text{ for all } i \in \{1, \ldots, n\},$$

where $e_i \in \mathbb{R}^n$ is the i-th canonical basis vector that selects the i-th column from W, and $\sigma_W \in (0,\ 1)$ is the target degree of atom sparseness.

In the situation where the learning samples resemble image patches, the atoms can also be restricted to fulfill structure sparseness using a low-rank approximation. Suppose the image patches possess $p_h \times p_w$ pixels, then the following projection can be carried out after each learning epoch:

$$W^{\text{new}} e_i := \left(\operatorname{vec} \circ \Pi_{\kappa_W} \circ \operatorname{vec}^{-1}_{p_h \times p_w}\right)(W e_i) \text{ for all } i \in \{1, \ldots, n\}.$$

Here, $\kappa_W \in \{1, \ldots, \min\{p_h, p_w\}\}$ denotes the target rank of each atom when reshaped to $p_h \times p_w$ pixels. When the dictionary is used to process images through convolution, small values of κ_W are beneficial since computations can then be speeded up considerably. For example, if $\kappa_W = 1$ each atom can be expressed as outer product of vectors from $\mathbb{R}^{p_h}$ and $\mathbb{R}^{p_w}$, respectively. The convolutional kernel is separable in this case, and two one-dimensional convolutions lead to the same result as one two-dimensional convolution, but require only a fraction of operations (Hoggar 2006).

3.3 Learning Algorithm

In the previous sections, a simple Hebbian-like learning rule was derived which depends on abstract inference models. The core of the inference models is the sparseness-enforcing projection operator discussed in Sect. 2, which guarantees that the code words always attain a sparseness degree easily controllable by the user. On presentation of a learning sample x, an update step of the dictionary W hence consists of

(a) Determining the filter responses ($u := W^T x$),
(b) Feeding the filter responses through the inference model involving a sparseness projection ($h := I(u)$),
(c) Computation of the approximation to the original learning sample (Wh) using the linear generative model,
(d) Assessing the similarity between the input sample and its approximation ($\rho(Wh,\ x)$) and the similarity measure's gradient ($g := \rho'(Wh,\ x)$), and eventually
(e) Adapting the current dictionary using a rank-1 update ($W^{\text{new}} := W - \eta \cdot gh^T$).

All these intermediate steps can be carried out efficiently. The sparseness projection can be computed using few resources, and neither its gradient nor the gradient of the entire inference model is required. After each learning epoch it is possible to impose additional constraints on the atoms with atom-wise projections. In doing so, several aspects of the data set structure can be made more explicit since the atoms are then forced to become more simple feature detectors.

An example of an individual learning epoch is given in Algorithm 1, where the topographic inference model and low-rank structure sparseness are used. In addition, the explicit expressions for the correlation coefficient and its gradient are given (Thom and Palm 2013, Proposition 40). The outcome of this parameterization applied to patches from natural images is depicted in Fig. 9b and will be discussed in Sect. 4.2.3.

The efficiency of the learning algorithm can be improved with simple modifications described in the following. To reduce the learning time compared to when a randomly initialized dictionary was used, the dictionary should be initialized by samples randomly chosen from the learning set (Mairal et al. 2009a; Skretting and Engan 2010; Coates and Ng 2011; Thom and Palm 2013). When large learning sets with millions of samples are available, stochastic gradient descent has been proven to result in significantly faster optimization progress compared to when the degrees of freedom are updated only once for each learning epoch (Wilson and Martinez 2003; Bottou and LeCun 2004). The step size schedule $\eta(\nu) := \eta_0/\nu$, where $\eta_0 > 0$ denotes the initial step size and $\nu \in \mathbb{N}\backslash\{0\}$ is the learning epoch counter, is beneficial when the true gradient is not available but rather an erroneous estimate (Bertsekas 1999). After each learning

Algorithm 1: One epoch of the Easy Dictionary Learning algorithm, illustrated for the case of a topographic inference model and low-rank structure sparseness

```
Input: The algorithm accepts the following parameters:
  – An existing dictionary $W \in \mathbb{R}^{d\times n}$ with $n$ atoms.
  – Random access to a learning set with samples from $\mathbb{R}^d$
    through a function "pick_sample".
  – Step size for update steps $\eta \in \mathbb{R}_{>0}$.
  – Number of samples to be presented $M_{\text{epoch}} \in \mathbb{N}$.
  – For the topographic inference model:
      – Target degree of dictionary sparseness $\sigma_H \in (0,\ 1)$.
      – Topography matrix $G \in \mathbb{R}^{n\times n}$ describing the interaction
        between adjacent atoms.
  – To obtain low-rank structure sparseness:
      – Number of pixels $p_h \times p_w$ so that $d = p_h p_w$.
      – Target rank $\kappa_W \in \{1, \ldots, \min\{p_h,\ p_w\}\}$.
Output: Updated dictionary $W \in \mathbb{R}^{d\times n}$.
   // Present samples and adapt dictionary.
1  repeat $M_{\text{epoch}}$ times
       // Pick a random sample.
2      $x :=$ pick_sample() $\in \mathbb{R}^d$;
       // (a) Compute filter responses.
3      $u := W^T x \in \mathbb{R}^n$;
       // (b) Evaluate topographic inference
          model.
4      $h := \Pi_{\sigma_H}(Gu) \in \mathbb{R}^n$;
       // (c) Compute approximation to $x$.
5      $\tilde{x} := Wh \in \mathbb{R}^d$;
       // (d) Compute correlation coefficient
          $\rho_{\tilde{x},\,x} := \rho(Wh,\ x)$ and its gradient $g$.
          Here, $e \in \{1\}^d$ is the all-ones vector.
6      $\lambda := \|\tilde{x}\|_2^2 - 1/d \cdot \left(e^T \tilde{x}\right)^2 \in \mathbb{R}$;
7      $\mu := \|x\|_2^2 - 1/d \cdot \left(e^T x\right)^2 \in \mathbb{R}$;
8      $\rho_{\tilde{x},\,x} := \left(\tilde{x}^T x - 1/d \cdot \left(e^T \tilde{x}\right)\left(e^T x\right)\right) / \sqrt{\lambda\mu} \in \mathbb{R}$;
9      $g := \frac{1}{\sqrt{\lambda\mu}}\left(x - e^T x/d \cdot e\right) - \frac{\rho_{\tilde{x},\,x}}{\lambda}\left(\tilde{x} - e^T \tilde{x}/d \cdot e\right) \in \mathbb{R}^d$;
       // (e) Adapt dictionary with rank-1
          update.
10     $W := W - \eta \cdot gh^T \in \mathbb{R}^{d\times n}$;
11 end
   // Atom-wise projection for low-rank
      filters.
12 for $i := 1$ to $n$ do
       // Extract $i$-th atom and normalize.
13     $w := We_i/\|We_i\|_2 \in \mathbb{R}^d$;
       // Ensure atom has rank $\kappa_W$ when reshaped
          to $p_h \times p_w$ pixels.
14     $w := \left(\text{vec} \circ \Pi_{\kappa_W} \circ \text{vec}^{-1}_{p_h \times p_w}\right)(w) \in \mathbb{R}^d$;
       // Store modified atom back in
          dictionary.
15     $We_i := w$;
16 end
```

epoch, all the atoms of the dictionary are normalized to unit scale. Since the learning rule is Hebbian-like, this prevents the atoms from growing arbitrarily large or becoming arbitrarily small (Hyvärinen et al. 2009). This normalization step is also common in a multitude of alternative dictionary learning algorithms (Kreutz-Delgado et al. 2003; Hoyer 2004; Mairal et al. 2009a; Skretting and Engan 2010).

Since the dictionary is modified with a rank-1 update where one of the factors is the result of an inference model and hence sparse, only the atoms that induce significant filter responses are updated. This may result in atoms that are never updated when the target sparseness degree σ_H is large. This behavior can be alleviated by adding random noise to the inference model's output prior to updating the dictionary, which forces all the atoms to be updated. We used random numbers sampled from a zero-mean normal distribution, where the standard deviation was multiplicatively annealed after each learning epoch. As optimization proceeds, the atoms are well-distributed in sample space such that the lifetime sparseness is approximately equal to the population sparseness (Willmore and Tolhurst 2001), and randomness is not needed anymore.

4 Experimental Results

This section reports experimental results on the techniques proposed in this paper. We first evaluate alternative solvers for the sparseness projection's auxiliary function and show that our algorithm for the sparseness-enforcing projection operator is significantly faster than previously known methods. We then turn to the application of the projection in the Easy Dictionary Learning algorithm. First, the morphology of dictionaries learned on natural image patches is analyzed and put in context with previous methods. We further show that the resulting dictionaries are well-suited for the reproduction of entire images. They achieve a reproduction quality equivalent to dictionaries trained with alternative, significantly slower algorithms. Eventually, we analyze the performance of the dictionaries when employed for the image denoising task and find that, analogous to the reproduction experiments, no performance degradation can be observed.

4.1 Performance of Sparseness-Enforcing Projections

Our proposed algorithm for sparseness projections was implemented as C++ program using the GNU Scientific Library (Galassi et al. 2009). We first analyzed whether solvers other than bisection for locating the zero of the auxiliary function would result in an improved performance. Since Ψ is differentiable except for isolated points and its derivatives can be computed quite efficiently, Newton's method and Halley's method are straightforward to apply. We

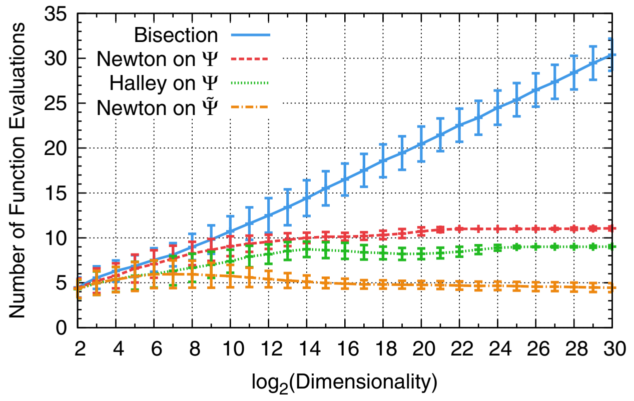


Fig. 4 Number of auxiliary function evaluations needed to find the final interval for the projection onto T using four different solvers. *The error bars* indicate ± 1 standard deviation distance from the mean value. Since Newton's method applied to $\tilde{\Psi}$ is consistently outperforming the other solvers, it is the method of choice for all practical applications

further verified whether Newton's method applied to a slightly transformed variant of the auxiliary function,

$$\tilde{\Psi}\colon\ [0,\ x_{\max}) \to \mathbb{R},\quad \alpha \mapsto \frac{\|\max(x - \alpha \cdot e,\ 0)\|_1^2}{\|\max(x - \alpha \cdot e,\ 0)\|_2^2} - \frac{\lambda_1^2}{\lambda_2^2},$$

where the minuend and the subtrahend were squared, would behave more efficiently. The methods based on derivatives were additionally safeguarded with bisection to guarantee new positions are always located within well-defined intervals (Press et al. 2007). This impairs the theoretical property that only a constant number of steps is required to find a solution, but in practice a significantly smaller number of steps needs to be made compared to plain bisection.

To evaluate which solver would provide maximum efficiency in practical scenarios, one thousand random vectors for each problem dimensionality $n \in \{2^2, \ldots, 2^{30}\}$ were sampled and the corresponding sparseness-enforcing projections for $\sigma^* := 0.90$ were computed using all four solvers. We have used the very same random vectors as input for all solvers, and counted the number of times the auxiliary function needed to be evaluated until the solution was found. The results of this experiment are depicted in Fig. 4.

The number of function evaluations required by plain bisection grew about linearly in $\log(n)$. Because the expected minimum difference of two distinct entries from a random vector gets smaller when the dimensionality of the random vector is increased, the expected number of function evaluations bisection needs increases with problem dimensionality. In either case, the length of the interval that has to be found is always bounded from below by the machine precision such that the number of function evaluations with bisection is bounded from above regardless of n.

The solvers based on the derivative of Ψ or $\tilde{\Psi}$, respectively, always required less function evaluations than

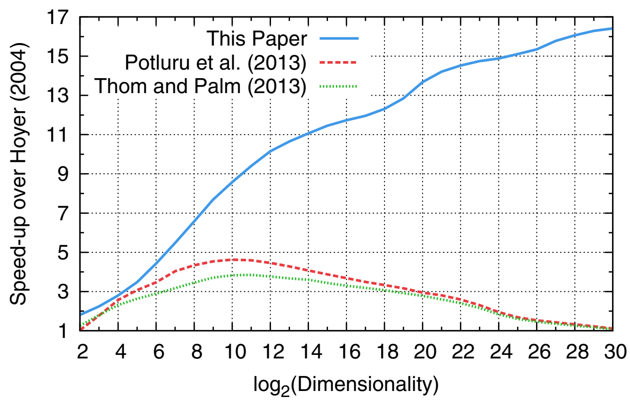


Fig. 5 Speed-ups relative to the original algorithm of Hoyer (2004) obtained using alternative algorithms. The linear time algorithm proposed in this paper is far superior in terms of execution time. The time complexity of the competing methods is at least quasi-linear, which becomes noticeable especially for large problem dimensionalities

bisection. They exhibited a growth clearly sublinear in $\log(n)$. For $n = 2^{30} \approx 10^9$, Newton's method required 11 function evaluations in the mean, Halley's method needed 9 iterations, and Newton's method applied to $\tilde{\Psi}$ found the solution in only 4.5 iterations. Therefore, in practice always the latter solver should be employed.

In another experiment, the time competing algorithms require for computing sparseness projections on a real computing machine was measured for a comparison. For this, the algorithms proposed by Hoyer (2004); Potluru et al. (2013) and Thom and Palm (2013) were implemented using the GNU Scientific Library (Galassi et al. 2009) by means of C++ programs. An Intel Core i7-4960X processor was used and all algorithms were run in a single-threaded environment. Random vectors were sampled for each problem dimensionality $n \in \{2^2, \ldots, 2^{30}\}$ and initially projected to attain a sparseness of 0.50 with respect to Hoyer's σ. This initial projection better reflects the situation in practice where not completely random vectors have to be processed. Next, all four algorithms were used to compute projections with a target sparseness degree of $\sigma^* := 0.90$ and their run time was measured. The original algorithm of Hoyer (2004) was the slowest, so by taking the ratio of the run times of the other algorithms to the run time of that slowest algorithm the relative speed-up was obtained.

Figure 5 visualizes the results of this experiment. For all tested problem dimensionalities, the here proposed linear time algorithm dominated all previously described methods. While the speed-up of the algorithms of Potluru et al. (2013) and Thom and Palm (2013) relative to the original algorithm of Hoyer (2004) were already significant, especially for small to medium dimensionalities, they got relatively slow for very large vectors. This is not surprising, because both methods start with sorting the input vector and have to store that permutation to be undone in the end.

The algorithm proposed in this paper is based on root-finding of a monotone function and requires no sorting. Only the left and right neighbors of a scalar in a vector have to be found. This can be achieved by scanning linearly through the input vector, which is particularly efficient when huge vectors have to be processed. For $n = 2^{30}$, the proposed algorithm was more than 15 times faster than the methods of Hoyer (2004), Potluru et al. (2013) and Thom and Palm (2013). Because of this appealing asymptotic behavior, there is now no further obstacle to applying smooth sparseness methods to large-scale problems.

4.2 Data Set Analysis with EZDL

We used the Easy Dictionary Learning algorithm as a data analysis tool to visualize the structure of patches from natural images under different aspects, which facilitates qualitative statements on the algorithm performance. For our experiments, we used the McGill calibrated color image database (Olmos and Kingdom 2004), where the images had either 768×576 pixels or 576×768 pixels. The images were desaturated (SMPTE RP 177-1993, Annex B.4) and quantized to 8-bit precision. We extracted 10000 patches with 16×16 pixels from each of the 314 images of the "Foliage" collection. The patches were extracted at random positions. Patches with vanishing variance were omitted since they carry no information. In total, we obtained 3.1 million samples for learning.

We learned dictionaries on the raw pixel values of this learning set and on whitened image patches. The learning samples were normalized to zero mean and unit variance as the only pre-processing for the raw pixel experiments. The whitened data was obtained using the canonical pre-processing from Section 5.4 of Hyvärinen et al. (2009) with 128 principal components. EZDL was applied using each proposed combination of inference model and atom constraints. We presented 30000 randomly selected learning samples in each of one thousand epochs, which corresponds to about ten sweeps through the entire learning set. The initial step size was set to $\eta_0 := 1$.

A noisy version of the final dictionary could already be observed after the first epoch, demonstrating the effectiveness of EZDL for quick analysis. Since the optimization procedure is probabilistic in the initialization and selection of training samples, we repeated the training five times for each parameter set. We observed only minor variations between the five dictionaries for each parameter set.

In the following, we present details of dictionaries optimized on raw pixel values and analyze their filter characteristics. Then, we consider dictionaries learned with whitened data and dictionaries with separable atoms trained on raw

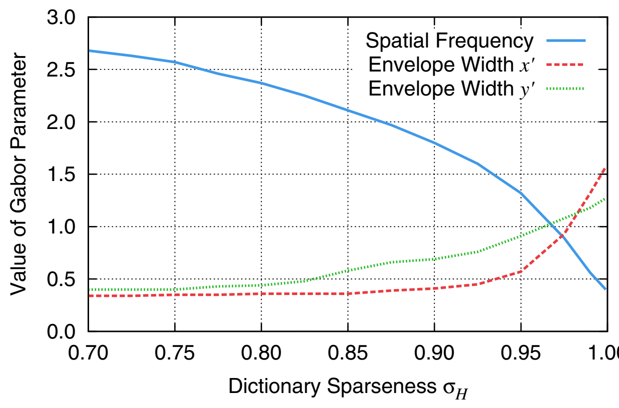


Fig. 6 Mean values of three influential parameters of Gabor functions fitted to the atoms of the dictionaries learned with EZDL, in dependence on the target degree of dictionary sparseness σ_H. An increase in sparseness results in filters with a reduced spatial frequency, but with a significant increase in the width of the Gaussian envelope

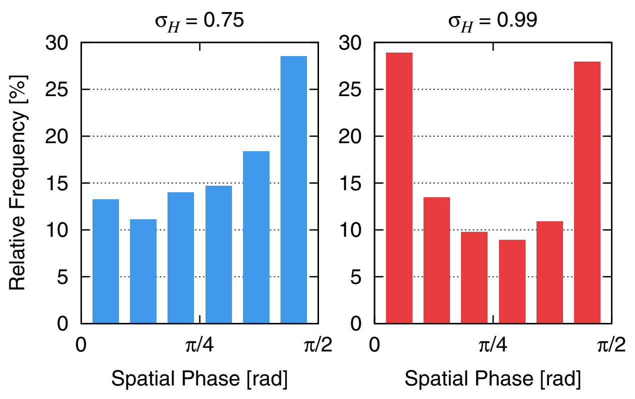


Fig. 7 Histograms of the spatial phase of fitted Gabor functions for $\sigma_H = 0.75$ (*left*) and $\sigma_H = 0.99$ (*right*). Low sparseness yields dictionaries where odd-symmetric filters dominate, whereas high dictionary sparseness results in a significant amount of both even-symmetric and odd-symmetric filters

pixel values and discuss the underlying reasons for their special morphology.

4.2.1 Raw Pixel Values

Using the ordinary inference model, we trained two-times overcomplete dictionaries with $n := 256$ atoms on the normalized pixel values, where the dictionary sparseness degree was varied between 0.700 and 0.999. This resulted in the familiar appearance of dictionaries with Gabor-like filters, which resemble the optimal stimulus of visual neurons (Olshausen and Field 1996, 1997). While for small values of σ_H the filters were mostly small and sharply bounded, high sparseness resulted in holistic and blurry filters.

We used the methods developed by Jones and Palmer (1987) and Ringach (2002) for a more detailed analysis. In doing so, a two-dimensional Gabor function as defined in Equation (1) of Ringach (2002), that is a Gaussian envelope multiplied with a cosine carrier wave, was fit to each dictionary atom using the algorithm of Nelder and Mead (1965). We verified that these Gabor fits accurately described the atoms, which confirmed the Gabor-like nature of the filters.

The differences in the dictionaries due to varying sparseness degrees became apparent through analysis of the parameters of the fitted Gabor functions. Figure 6 shows the mean values of three influential Gabor parameters in dependence on σ_H. All parameters change continuously and monotonically with increasing sparseness. The spatial frequency, a factor in the cosine wave of the Gabor function, constantly decreases for increasing dictionary sparseness. The width of the Gaussian envelope, here broken down into the standard deviation in both principal orientations x' and y', is monotonically increasing. The envelope width in y' direction increases earlier, but in the end the envelope width in x' direction is larger.

It can be concluded that more sparse code words result in filters with lower frequency but larger envelope. Since an increased sparseness reduces the model's effective number of degrees of freedom, it prevents the constrained dictionaries from adapting precisely to the learning data. Similar to principal component analysis, low-frequency atoms here minimize the reproduction error best when only very few effective atoms are allowed (Hyvärinen et al. 2009; Olshausen and Field 1996).

Histograms of the spatial phase of the filters, which is an additive term in the cosine wave of the Gabor function, are depicted in Fig. 7. For $\sigma_H = 0.75$, there is a peak at $\pi/2$ which corresponds to odd-symmetric filters (Ringach 2002). The distribution is clearly bimodal for $\sigma_H = 0.99$, with peaks at 0 and $\pi/2$ corresponding to even-symmetric and odd-symmetric filters, respectively. While the case of small σ_H matches the result of ordinary sparse coding, the higher dictionary sparseness results in filters with the same characteristics as the optimal stimulus of macaque visual neurons (Ringach 2002).

This analysis proves that EZDL's minimalistic learning rule is capable of generating biologically plausible dictionaries, which constitute a particularly efficient image coding scheme. To obtain dictionaries with diverse characteristics, it is enough to adjust the target degree of dictionary sparseness on a normalized scale. A major component in the learning algorithm is the sparseness projection, enforcing local competition among the atoms (Rozell et al. 2008) due to its absolute order-preservation property (Thom and Palm 2013, Lemma 12).

4.2.2 Whitened Image Patches

Whitening as a pre-processing step helps to reduce sampling artifacts and decorrelates the input data (Hyvärinen et al.

2009). It also changes the intuition of the similarity measure in EZDL's objective function, linear features rather than single pixels are considered where each feature captures a multitude of pixels in the raw patches. This results in differences in the filter structure, particularly in the emergence of more low-frequency filters.

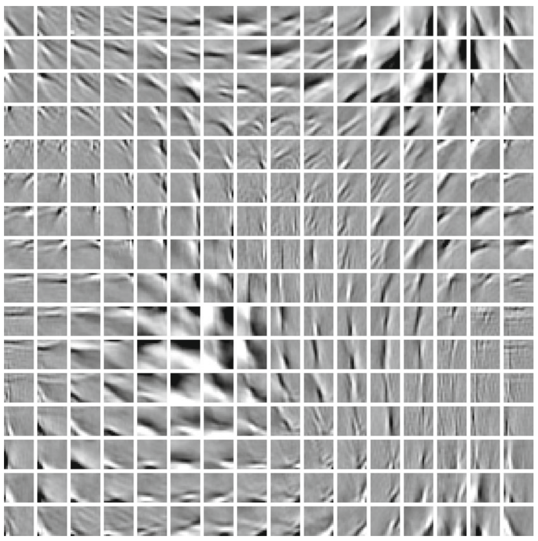
The dictionary depicted in Fig. 8a was learned using the topographic inference model with average-pooling of 3×3 neighborhoods. We set the dictionary sparseness degree to $\sigma_H := 0.85$ and the number of atoms to $n := 256$, arranged on a 16×16 grid. The dictionary closely resembles those gained through topographic independent component analysis (Hyvärinen et al. 2001, 2009) and invariant predictive sparse decomposition (Kavukcuoglu et al. 2009). It should be noted that here the representation is two times overcomplete due to the whitening procedure. Overcomplete representations are not inherently possible with plain independent component analysis (Bell and Sejnowski 1997; Hyvärinen et al. 2009) which limits its expressiveness, a restriction that does not hold for EZDL.

The emergence of topographic organization can be explained by the special design of the topographic inference model. The pooling operator acts as spatial low-pass filter on the filter responses, so that each smoothed filter response carries information on the neighboring filter responses. Filter response locality is retained after the sparseness projection, hence adjacent atoms receive similar updates with the Hebbian-like learning rule. Hence, there are only small differences between filters within the same vicinity. The learning process is similar to that of self-organizing maps (Kohonen 1990), where only the atom with the maximum filter response and the atoms in its direct surrounding are updated. However, EZDL simultaneously updates multiple clusters since the sparse code words laid out on the grid are multimodal.

Further, it is notable that we achieved a topographic organization merely through a linear operator G by simple average-pooling. This stands in contrast to the discussion from Bauer and Memisevic (2013) where the necessity of a nonlinearity such as multiplicative interaction or pooling with respect to the Euclidean norm was assumed. Our result that a simple linear operator plugged into the ordinary inference model already produces smooth topographies proves that linear interactions between the atoms are sufficient despite their minimality.

Figure 8b shows a dictionary obtained with the rank-1 topographic inference model, using a sparseness degree of $\sigma_H := 0.85$ and $n := 256$ filters on a 16×16 grid. Here, the sparse code words reshaped to a matrix are required to have unit rank which results in a specially organized filter layout. For example, rows three and four almost exclusively contain low-frequency filters, and the filters in the sixth column are all oriented vertically. The grouping of similar atoms into the same rows and columns is related to independent subspace analysis, which yields groups of Gabor-like filters where all filters in a group have approximately the same frequency and orientation (Hyvärinen and Hoyer 2000).

The rank-1 topographic inference model guarantees that code words can be expressed as the dyadic product of two vectors, where the factors themselves are sparsely populated because the code words are sparse. This causes the code words to possess a sparse rectangular structure when reshaped to account for the grid layout, that is non-vanishing activity is always concentrated in a few rows and columns. The Hebbian-like learning rule therefore induces similar updates to atoms located in common rows or columns, which explains the obtained group layout.


(a) Topographic inference model

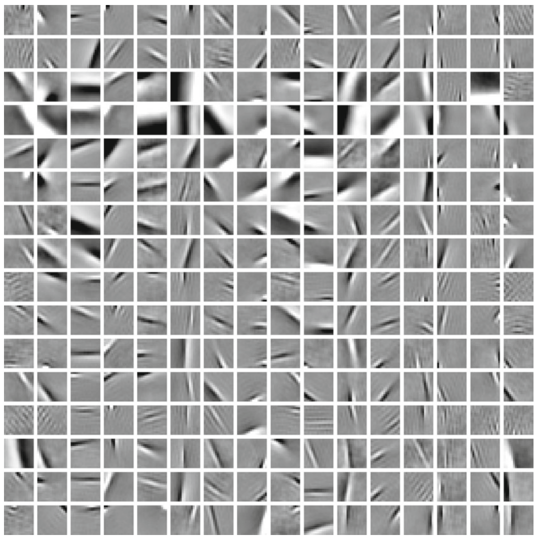
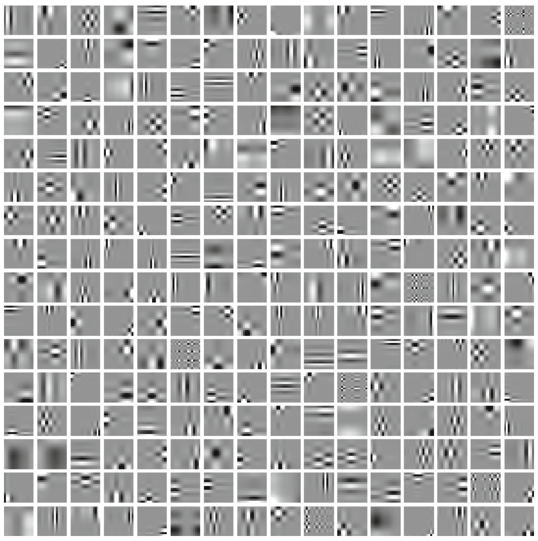
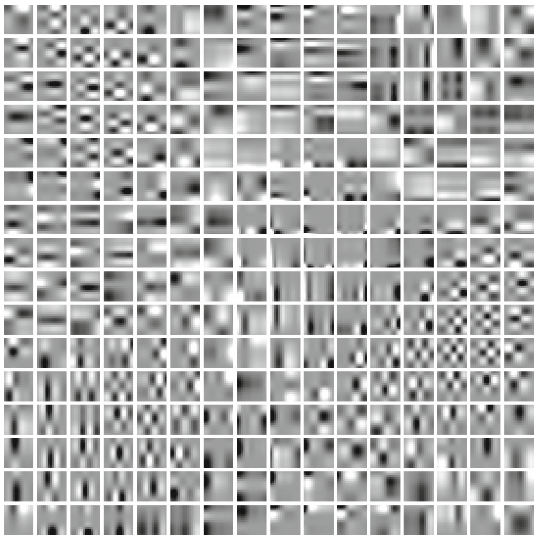

(b) Rank-1 topographic inference model

Fig. 8 Two times overcomplete dictionaries trained with EZDL on whitened natural image patches



(a) Ordinary inference model



(b) Topographic inference model

Fig. 9 Dictionaries trained with EZDL on raw pixels, the atoms were replaced by their best rank-1 approximation after each epoch

4.2.3 Rank-1 Filters on Raw Pixel Values

Enforcing the filters themselves to have rank one by setting $\kappa_W := 1$ resulted in bases similar to discrete cosine transform (Watson 1994). This was also observed recently by Hawe et al. (2013) who considered a tensor decomposition of the dictionary, and by Rigamonti et al. (2013) who minimized the Schatten 1-norm of the atoms. Note that EZDL merely replaces the atoms after each learning epoch by their best rank-1 approximation. The computational complexity of this operation is negligible compared to an individual learning epoch using tens of thousands of samples and hence does not slow down the actual optimization.

If no ordering between the filters was demanded, a multitude of checkerboard-like filters and vertical and horizontal contrast bars was obtained, see Fig. 9a for a dictionary with $n := 256$ atoms and a sparseness degree of $\sigma_H := 0.85$. Atoms with diagonal structure cannot be present at all in such constrained dictionaries because diagonality requires a filter rank greater than one. Although all filters were paraxial due to the rank-1 constraint, they still resembled contrast fields because of their grating-like appearance. This is due to the representation's sparseness, which is known to induce this appearance.

A similar filter morphology was obtained with a topographic filter ordering using a 16×16 grid, though the filters were blurrier, as shown in Fig. 9b. Here, checkerboard-like filters are located in clusters, and filters with vertical and horizontal orientation are spatially adjacent. This is as expected, because with a sparse topography there are only a few local blobs of active entries in each code word, causing similar atoms to be grouped together and dissimilar atoms to be distant from each other. In the space of rank-1 filters, there can be either vertical or horizontal structures, or checkerboards combining both principal orientations.

The lack of fine details can be explained by the restriction of topographic organization, which reduces the effective number of degrees of freedom. Analogously to the situation in Sect. 4.2.1 where the dictionary sparseness was increased, the reproduction error can be reduced by a large amount using few filters with low frequencies. For the same reason, there are few such checkerboard-like filters: Minimization of the reproduction error is first achieved with principal orientations before checkerboards can be used to encode details in the image patches.

In conclusion, these results show that the variety of the dictionaries produced by EZDL can be vast simply by adapting easily interpretable parameters. The algorithm covers and reproduces well-known phenomena from the literature, and allows a precise visualization of a data set's structure. A first impression of the final dictionaries can already be obtained after one learning epoch, which takes no more than one second on a modern computer.

4.3 Experiments on the Reproduction of Entire Images

It was demonstrated in Sect. 4.2 that the Easy Dictionary Learning algorithm produces dictionaries that correspond to efficient image coding schemes. We now analyze the suitability of EZDL dictionaries trained on raw pixel values for the reproduction of entire images. This is a prerequisite for several image processing tasks such as image enhancement and compression, since here essentially the same optimization problem should be solved as during reproduction with a given dictionary.

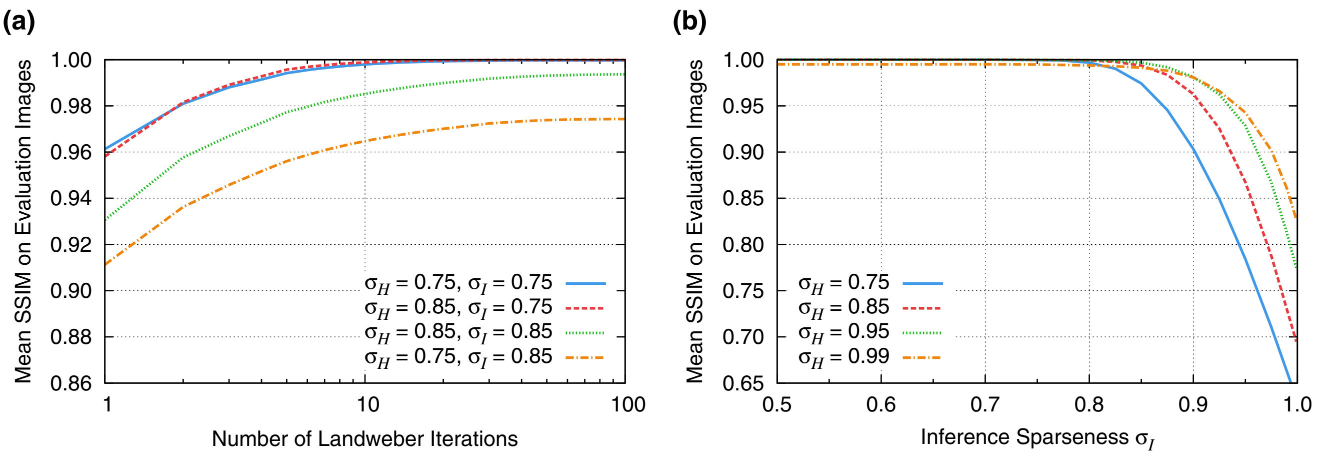


Fig. 10 Results on the reproduction of entire images using Easy Dictionary Learning dictionaries. **a** While one Landweber iteration suffices for learning, better reproduction performance is yielded if the Landweber procedure is run until convergence. **b** The relation between the dictionary sparseness σ_H and the inference sparseness σ_I is influential. Hardly any information loss can be observed for $\sigma_I < \sigma_H$, while the case $\sigma_I > \sigma_H$ shows dictionaries should be trained for very sparse code words when demanding such a sparseness for reproduction

Our analysis allows quantitative statements as the original images and their reproduction through sparse coding can be numerically compared on the signal level. Further, the EZDL dictionaries are compared with the results of the Online Dictionary Learning algorithm by Mairal et al. (2009a) and the Recursive Least Squares Dictionary Learning Algorithm by Skretting and Engan (2010) in terms of reproduction quality and learning speed.

The evaluation methodology was as follows. First, dictionaries with different parameter sets were trained on 3.1 million 8×8 pixels natural image patches extracted from the "Foliage" collection of the McGill calibrated color image database (Olmos and Kingdom 2004). As in Sect. 4.2, 10000 patches were picked from random positions from each of the desaturated and quantized images. All patches were normalized to zero mean and unit variance before training. The dictionaries were designed for a four times overcomplete representation, they had $n := 256$ atoms for samples with 64 pixels. After training, dictionary quality was measured using the 98 images of the "Flowers" collection from the same image database. This evaluation set was disjoint from the learning set.

Each single image was subsequently divided into non-overlapping blocks with 8×8 pixels. All blocks were normalized to zero mean and unit variance, and then the optimized dictionaries were used to infer a sparse code word for each block. The resulting code word was then fed through a linear generative model using the currently investigated dictionary, and the mean value and variance were restored to match that of the original block.

Sparse code word inference was achieved with a projected Landweber procedure (Bredies and Lorenz 2008), essentially the same as projected gradient descent starting with the null vector. Using the sparseness-enforcing projection operator after each iteration, the code words were tuned to attain an *inference sparseness* σ_I, which need not necessarily be equal to the sparseness degree used for dictionary learning. The correlation coefficient was used as the similarity measure for inference to benefit from invariance to shifting and scaling. Automatic step size control was carried out with the bold driver heuristic (Bishop 1995). We note that due to the representation theorem on the sparseness projection this process can also be understood as iterative soft-thresholding (Bredies and Lorenz 2008).

A reproduced image is yielded by applying this procedure to all distinct blocks of the original image using a single dictionary. The deviation between this output image and the original image was assessed with the SSIM index (Wang and Bovik 2009), which yielded qualitatively similar results as the peak signal-to-noise ratio. The SSIM index is, however, normalized to values between zero and one, and it respects the local structure of images since it examines 11×11 pixels neighborhoods through convolution. Because it measures the visual similarity between images, it is more intuitive than measures based on the pixel-wise squared error (Wang and Bovik 2009). This evaluation method yielded one number from the interval $[0,\ 1]$ for each image and dictionary. Each parameterization for dictionary learning was used to train five dictionaries to compensate probabilistic effects, and here the mean of the resulting 490 SSIMs is reported.

4.3.1 EZDL Results

Figure 10 visualizes the results obtained for dictionaries produced by Easy Dictionary Learning using dictionary sparseness degrees $\sigma_H \in \{0.75,\ 0.85,\ 0.95,\ 0.99\}$. One thousand learning epochs were carried out, presenting 30000 samples in each epoch, and using an initial step size of $\eta_0 := 1$. During dictionary learning, only the first trivial iteration of a Landweber procedure is carried out for sparse code word inference by computing $h := \Pi_{\sigma_H}(W^T x)$ for the ordinary inference model.

We first analyzed the impact of carrying out more Landweber iterations during the image reproduction phase, and the effect of varying the inference sparseness degree σ_I. A huge performance increase can be obtained by using more than one iteration for inference (Fig. 10a). Almost the optimal performance is achieved after ten iterations, and after one hundred iterations the method has converged. For $\sigma_I = 0.75$, the performance of dictionaries with $\sigma_H = 0.75$ and $\sigma_H = 0.85$ is about equal yielding the maximum SSIM of one. This value indicates that there is no visual difference between the reproduced images and the original images.

When the inference sparseness σ_I is increased to 0.85, a difference in the choice of the dictionary sparseness σ_H becomes noticeable. For $\sigma_H = 0.85$, performance already degrades, and the degradation is substantial if $\sigma_H = 0.75$. It is more intuitive when σ_H and σ_I are put in relation. Almost no information is lost in the reproduction by enforcing a lower inference sparseness than dictionary sparseness ($\sigma_I < \sigma_H$). Performance is worse if $\sigma_I > \sigma_H$ which is plausible because the dictionary was not adapted for a higher sparseness degree. In the natural case $\sigma_I = \sigma_H$ the performance mainly depends on their concrete value, where higher sparseness results in worse reproduction capabilities.

To further investigate this behavior, we set the number of Landweber iterations to 100 and varied σ_I smoothly in the interval $[0.50,\ 1.0)$ for the four different dictionary sparseness degrees. The results of this experiment are depicted in Fig. 10b. For $\sigma_I \leq 0.80$ there is hardly any difference in the reproduction quality irrespective of the value of σ_H. The difference when training dictionaries with different sparseness degrees first becomes visible for large values of σ_I. Performance is here better using dictionaries where very sparse code words were demanded during learning. Hence, tasks that require high code word sparseness should use dictionaries trained with high values of the dictionary sparseness.

4.3.2 *Comparison with Alternative Dictionary Learning Algorithms*

For a comparison, we conducted experiments using the Online Dictionary Learning (ODL) algorithm of Mairal et al. (2009a) and the Recursive Least Squares Dictionary Learning Algorithm (RLS-DLA) by Skretting and Engan (2010). For inference of sparse code words, ODL minimizes the reproduction error under implicit sparseness constraints. RLS-DLA uses an external vector selection algorithm for inference, hence explicit constraints such as a target L_0 pseudo-norm can be demanded easily. Both algorithms update the dictionary after each sample presentation.

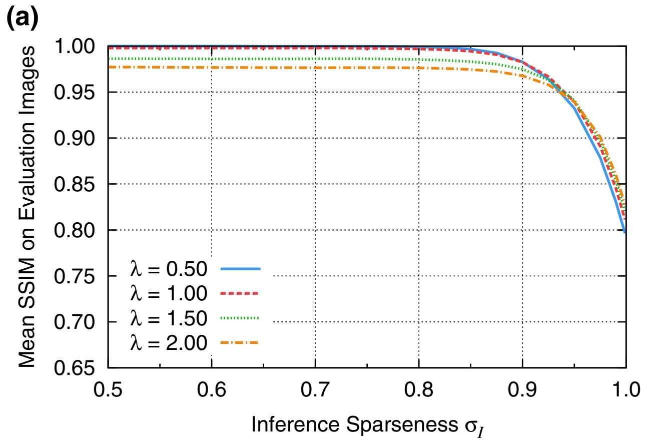
ODL does not require any step sizes to be adjusted. The crucial parameter for dictionary sparseness here is a number $\lambda \in \mathbb{R}_{>0}$, which controls the trade-off between reproduction capabilities and code word sparseness. We trained five dictionaries for each $\lambda \in \{0.50,\ 1.00,\ 1.50,\ 2.00\}$ using $n := 256$ atoms and presenting 30000 learning samples in each of one thousand epochs. Then, the same evaluation methodology as before with $\sigma_I \in [0.50,\ 1.0)$ was used to assess the reproduction capabilities. The results are shown in Fig. 11a. The choice of λ is not as influential compared to σ_H in EZDL. There are only small performance differences for $\sigma_I < 0.925$, and for $\sigma_I \geq 0.925$ hardly any difference can be observed. Similar to the EZDL experiments, large values of λ result in better performance for large σ_I and worse performance for small σ_I.

RLS-DLA provides a forgetting factor parameter which behaves analogously to the step size in gradient descent. We used the default forgetting factor schedule which interpolates between 0.99 and 1 using a cubic function over one thousand learning epochs. For inference, we chose an optimized Orthogonal Matching Pursuit variant (Gharavi-Alkhansari and Huang 1998) where a parameter $\zeta \in \mathbb{N}$ controls the number of non-vanishing coordinates in the code words. We trained five dictionaries with $n := 256$ atoms for each $\zeta \in \{4,\ 8,\ 16,\ 32\}$. Thirty thousand randomly drawn learning samples were presented in each learning epoch. The resulting reproduction performance is shown in Fig. 11b. Again, for large values of σ_I the dictionaries with the most sparse code words during learning perform best, that is those trained with small values of ζ.

To compare all three dictionary learning algorithms independent of the concrete dictionary sparseness, we took the mean SSIM value that belonged to the best performing dictionaries for each feasible value of σ_I. This can also be interpreted as using the convex hull of the results shown in Figs. 10b and 11. This yielded one curve for each dictionary learning algorithm, depicted in Fig. 12. There is only a minor difference between the curves over the entire range of σ_I. It can hence be concluded that the algorithms learn dictionaries which are equally well-suited for the reproduction of entire images. Although EZDL uses a very simple learning rule, this is sufficient enough to achieve a performance competitive with that of two state-of-the-art dictionary learning algorithms.

4.3.3 *Comparison of Learning Speed*

We moreover compared the learning speed of the methods and investigated the influence of the initial step size on the EZDL dictionaries. In doing so, we carried out reproduction experiments on entire images using dictionaries



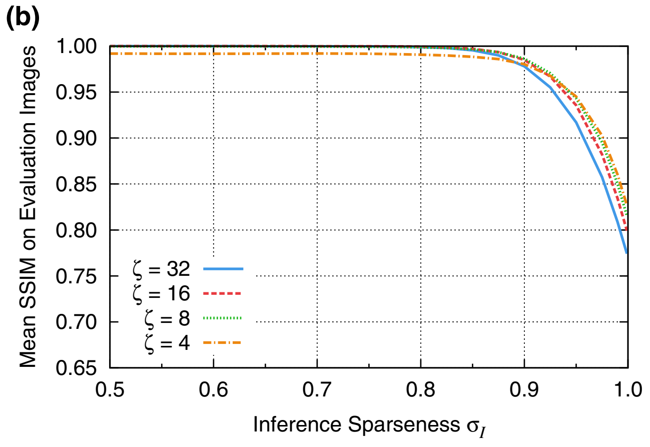


Fig. 11 Experimental results on image reproduction where alternative dictionary learning algorithms were used. The dictionaries behave similarly to Easy Dictionary Learning dictionaries if the dictionary sparseness parameter and the inference sparseness are varied. **a** Results of the Online Dictionary Learning (ODL) algorithm by Mairal et al. (2009a), where λ trades off between sparseness and reproduction capabilities in a model with implicit sparseness constraints. **b** Results of the Recursive Least Squares Dictionary Learning Algorithm (RLS-DLA) by Skretting and Engan (2010). Here, ζ denotes the target L_0 pseudo-norm of the code words for inference

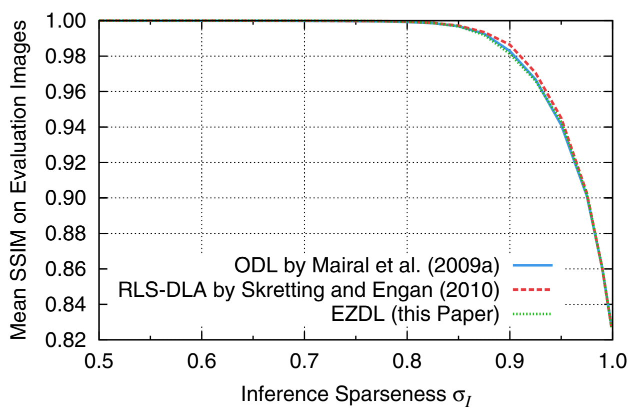


Fig. 12 Comparison of dictionary performance when the dependency on the dictionary sparseness degrees λ, ζ and σ_H was eliminated, for ODL, RLS-DLA and EZDL, respectively. All three algorithms produce dictionaries equally well-suited to the reproduction of entire images

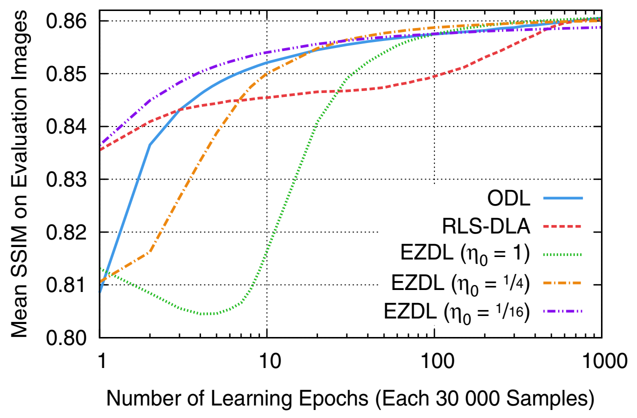


Fig. 13 Reproduction quality in dependence on the number of samples presented to Online Dictionary Learning, the Recursive Least Squares Dictionary Learning Algorithm, and Easy Dictionary Learning, where the initial step size η_0 was varied for the latter. All approaches achieved the same performance after 1,000 learning epochs

provided by the learning algorithms after certain numbers of learning epochs. In each of one thousand overall epochs, 30000 learning samples where input to all three algorithms. The dictionary sparseness parameters were set to $\sigma_H := 0.99$ for EZDL, to $\lambda := 2.00$ for Online Dictionary Learning, and to $\zeta := 4$ for the Recursive Least Squares Dictionary Learning Algorithm.

Timing measurements on an Intel Core i7-4960X processor have shown that one EZDL learning epoch takes approximately 30 % less time than a learning epoch of ODL. RLS-DLA was roughly 25 times slower than EZDL. Although the employed vector selection method was present as part of an optimized software library, only a fairly unoptimized implementation of the actual learning algorithm was available. Therefore, a comparison of this algorithm with regard to execution speed would be inequitable.

The inference sparseness was set to $\sigma_I := 0.99$ and one hundred Landweber iterations were carried out for sparse code word inference. The SSIM index was eventually used to assess the reproduction performance. Figure 13 visualizes the results of the learning speed analysis, averaged over all images from the evaluation set and five dictionaries trained for each parameterization to compensate probabilistic effects. Online Dictionary Learning is free of step sizes, dictionary performance increased instantly and constantly with each learning epoch. Only small performance gains could be observed after 100 learning epochs.

The performance of dictionaries trained with RLS-DLA was initially better than that of the ODL dictionaries, but improved more slowly. After a hundred epochs, however, it was possible to observe a significant performance gain. Although tweaking the forgetting factor schedule may have led to better early reproduction capabilities, RLS-DLA achieved a performance equivalent to that of ODL after 1000 epochs.

If EZDL's initial step size was set to unity, performance first degraded, but started to improve after the fifth epoch. After 100 epochs, the performance was identical to ODL's, and all three methods obtained equal reproduction capabilities after 1000 epochs. Reduction of the initial step size caused the dictionaries to perform better after few sample presentations. The quality of Online Dictionary Learning was achieved after 20 epochs for $\eta_0 = 1/4$, and for $\eta_0 = 1/16$ the EZDL dictionaries were always better in the mean until the 100th epoch. There is hardly any quality difference after 1000 epochs, a very small initial step size resulted however in a slightly worse performance.

The choice of the initial step size for EZDL is hence not very influential. Although $\eta_0 = 1$ caused small overfitting during the very first epochs, the dictionaries quickly recovered so that no significant difference in reproduction capabilities could be observed after 100 epochs. Since an EZDL epoch is 30 % faster than an Online Dictionary Learning epoch, our proposed method produces better results earlier on an absolute time scale.

4.4 Image Denoising Experiments

We have demonstrated in Sect. 4.3 that the dictionaries learned with Easy Dictionary Learning are as good as those obtained from the Online Dictionary Learning algorithm of Mairal et al. (2009a) and the Recursive Least Squares Dictionary Learning Algorithm by Skretting and Engan (2010) in terms of the reproduction quality of entire images. In a final experiment, we investigated the suitability of EZDL dictionaries for image denoising using the image enhancement procedure proposed by Mairal et al. (2009b).

This method carries out semi-local block matching and finds sparse code words by imposing a group sparseness penalty on the Euclidean reproduction error. In doing so, a pre-learned dictionary is used which explains the appearance of uncorrupted images and further helps to resolve ambiguities if block matching fails to provide large enough groups. A linear generative model is finally used to find estimators of denoised image patches from the sparse code words. This procedure is quite robust if the input data is noisy, since sparseness provides a strong prior which well regularizes this ill-posed inverse problem (Kreutz-Delgado et al. 2003; Foucart and Rauhut 2013).

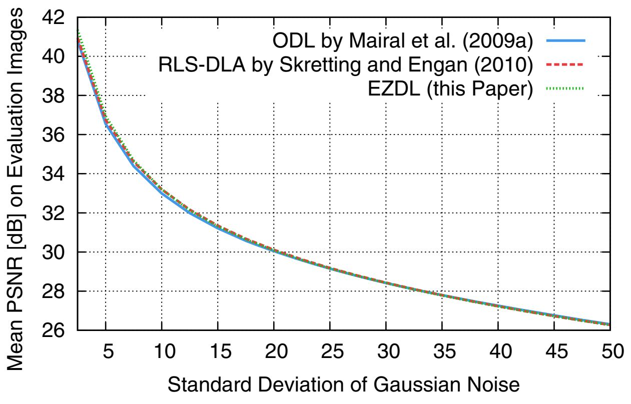


Fig. 14 Denoising performance in terms of the peak signal-to-noise ratio (PSNR) using the denoising method proposed by Mairal et al. (2009b). There is only a small performance difference between dictionaries trained with ODL, RLS-DLA and EZDL

The denoising approach of Mairal et al. (2009b) also provides the possibility of dictionary adaptation while denoising concrete input data. We did not use this option as it would hinder resilient statements on a dictionary's eligibility if it would be modified with another dictionary learning algorithm during denoising.

The methodology of the experiment was as follows. We used the four-times overcomplete dictionaries trained on the 8×8 pixels image patches from the "Foliage" collection of the McGill calibrated color image database (Olmos and Kingdom 2004) as models for uncorrupted images. The dictionary sparseness parameters were $\sigma_H := 0.99$ for EZDL, $\lambda := 2.00$ for ODL and $\zeta := 4$ for RLS-DLA. For evaluation, we used the 81 images of the "Animals" collection from the McGill database, and converted them to 8-bit grayscales as in the previous experiments. The images were synthetically corrupted with additive white Gaussian noise. Five noisy images were generated for each original image and each standard deviation from $\{2.5, 5.0, \ldots, 47.5, 50.0\}$.

These images were then denoised using the candidate dictionaries, where the window size for block matching was set to 32 pixels. Then, the similarity between the reconstructions and the original images was measured with the peak signal-to-noise ratio. We also evaluated the SSIM index, which led to qualitatively similar results.

The results are depicted in Fig. 14. Denoising performance degrades if the noise strength is increased. There is hardly any difference between the dictionaries trained with the three algorithms. The RLS-DLA and EZDL dictionaries perform slightly better for small synthetic noise levels, but this improvement is visually imperceptible. This result is not surprising, since Sect 4.3 demonstrated that all three learning algorithms produce dictionaries equally well-suited for the reproduction of entire images.

The denoising procedure of Mairal et al. (2009b) aims at reproduction capabilities as well, with the modification of employing noisy samples as input. Image enhancement and compression applications such as those proposed by Yang et al. (2010, 2012), Dong et al. (2011), Skretting and Engan (2011) and Horev et al. (2012) which also use problem formulations based on the reproduction error can hence be expected to benefit from more efficiently learned dictionaries as well.

5 Conclusions

This paper proposed the EZDL algorithm which features explicit sparseness constraints with respect to Hoyer's smooth sparseness measure σ. Pre-defined sparseness degrees are ensured to always be attained using a sparseness-enforcing projection operator. Building upon a succinct representation of the projection, we proved that the projection problem can be formulated as a root-finding problem. We presented a linear time and constant space algorithm for the projection which is superior to previous approaches in terms of theoretical computational complexity and execution time on real computing machines.

EZDL adapts dictionaries to measurement data through simple rank-1 updates. The sparseness projection serves as foundation for sparse code word inference. Due to the projection efficiency and since no complicated gradients are required, our proposed learning algorithm is significantly faster than even the optimized ODL algorithm. Topographic atom organization and atom sparseness can be realized with very simple extensions, allowing for versatile sparse representations of data sets. Its simplicity and efficiency does not hinder EZDL from producing dictionaries competitive with those generated by ODL and RLS-DLA in terms of the reproduction and denoising performance on natural images. Alternative image processing methods based on sparse representations rely on dictionaries subject to the same criteria, and can thus be expected to benefit from EZDL's advantages as well.

Acknowledgments The authors are grateful to Heiko Neumann, Florian Schüle, and Michael Gabb for helpful discussions. We would like to thank Julien Mairal and Karl Skretting for making implementations of their algorithms available. Parts of this work were performed on the computational resource bwUniCluster funded by the Ministry of Science, Research and Arts and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC. This work was supported by Daimler AG, Germany.

Appendix: Technical Details and Proofs for Section 2

This appendix studies the algorithmic computation of Euclidean projections onto level sets of Hoyer's σ in greater detail, and in particular proves the correctness of the algorithm proposed in Sect. 2.

For a non-empty subset $M \subseteq \mathbb{R}^n$ of the Euclidean space and a point $x \in \mathbb{R}^n$, we call

$$\operatorname{proj}_M(x) := \{y \in M \mid \|y - x\|_2 = \inf_{z \in M} \|z - x\|_2\}$$

the set of Euclidean projections of x onto M (Deutsch 2001). Since we only consider situations in which $\operatorname{proj}_M(x) = \{y\}$ is a singleton, we may also write $y = \operatorname{proj}_M(x)$.

Without loss of generality, we can compute $\operatorname{proj}_T(x)$ for a vector $x \in \mathbb{R}^n_{\geq 0}$ within the non-negative orthant instead of $\operatorname{proj}_S(x)$ for an arbitrary point $x \in \mathbb{R}^n$ to yield sparseness-enforcing projections, where T and S are as defined in Sect. 2. First, the actual scale is irrelevant as we can simply rescale the result of the projection (Thom and Palm 2013, Remark 5). Second, the constraint that the projection lies in the non-negative orthant $\mathbb{R}^n_{\geq 0}$ can easily be handled by flipping the signs of certain coordinates (Thom and Palm 2013, Lemma 11). Finally, all entries of x can be assumed non-negative with Corollary 19 from Thom and Palm (2013).

We note that T is non-convex because of the $\|s\|_2 = \lambda_2$ constraint. Moreover, $T \neq \emptyset$ for all target sparseness degrees $\sigma^* \in (0,\ 1)$ which we show here by construction (see also Remark 18 in Thom and Palm (2013) for further details): Let $\psi := \left(\lambda_1 - \sqrt{n\lambda_2^2 - \lambda_1^2}/\sqrt{n-1}\right)/n > 0$ and $\omega := \lambda_1 - (n-1)\psi > 0$, then the point $q := \sum_{i=1}^{n-1} \psi e_i + \omega e_n \in \mathbb{R}^n$ lies in T, where $e_i \in \mathbb{R}^n$ denotes the i-th canonical basis vector. Since all coordinates of q are positive, T always contains points with an L_0 pseudo-norm of n. If we had used the L_0 pseudo-norm to measure sparseness, then q would have the same sparseness degree as, for example, the vector with all entries equal to unity. If, however, σ^* is close to one, then there is only one large value ω in q and all the other entries equaling ψ are very small but positive. This simple example demonstrates that in situations where the presence of noise cannot be eliminated, Hoyer's σ is a much more robust sparseness measure than the L_0 pseudo-norm.

Representation Theorem

Before proving a theorem on the characterization of the projection onto T, we first fix some notation. As above, let $e_i \in \mathbb{R}^n$ denote the i-th canonical basis vector and let furthermore $e := \sum_{i=1}^n e_i \in \mathbb{R}^n$ be the vector where all entries are one. We note that if a point x resides in the non-negative orthant, then $\|x\|_1 = e^T x$. Subscripts to vectors denote the corresponding coordinate, except for e and e_i. For example, we have that $x_i = e_i^T x$. We abbreviate $\xi \in \mathbb{R}_{\geq 0}$ with $\xi \geq 0$ when it is clear that ξ is a real number. When $I \subseteq \{1, \ldots, n\}$ is an index set with $d := |I|$ elements, say $i_1 < \cdots < i_d$, then the unique matrix $V_I \in \{0, 1\}^{d \times n}$ with $V_I x = \left(x_{i_1}, \ \ldots, \ x_{i_d}\right)^T \in \mathbb{R}^d$ for all $x \in \mathbb{R}^n$ is called the *slicing operator*. A useful relation between the L_0 pseudo-norm, the Manhattan norm and the Euclidean norm is $\|x\|_2 \le \|x\|_1 \le \|x\|_0^{1/2}\,\|x\|_2 \le \sqrt{n}\,\|x\|_2$ for all points $x \in \mathbb{R}^n$.

We are now in a position to formalize the representation theorem:

Theorem 1 *Let $x \in \mathbb{R}^n_{\ge 0} \backslash T$ and $p := \operatorname{proj}_T(x)$ be unique. Then there is exactly one number $\alpha^* \in \mathbb{R}$ such that*

$$p = \beta^* \cdot \max\left(x - \alpha^* \cdot e,\ 0\right),$$

where $\beta^* := {}^{\lambda_2}/\|\max(x-\alpha^*\cdot e,\ 0)\|_2 > 0$ is a scaling constant. Moreover, if $I := \{i \in \{1,\dots,n\} | p_i > 0\} = \{i_1,\dots,i_d\}$, $d := |I|$ and $i_1 < \dots < i_d$, denotes the set of the d coordinates in which p does not vanish, then

$$\alpha^* = \frac{1}{d}\left(\|V_I x\|_1 - \lambda_1\sqrt{\frac{d\,\|V_I x\|_2^2 - \|V_I x\|_1^2}{d\lambda_2^2 - \lambda_1^2}}\right).$$

Proof It is possible to prove this claim either constructively or implicitly, where both variants differ in whether the set I of all positive coordinates in the projection can be computed from x or must be assumed to be known. We first present a constructive proof based on a geometric analysis conducted in Thom and Palm (2013), which contributes to deepening our insight into the involved computations. As an alternative, we also provide a rigorous proof using the method of Lagrange multipliers which greatly enhances the unproven analysis of (Potluru et al., 2013, Section 3.1).

We first note that when there are $\alpha^* \in \mathbb{R}$ and $\beta^* > 0$ so that we have $p = \beta^* \cdot \max(x - \alpha^* \cdot e,\ 0)$, then β^* is determined already through α^* because it holds that $\lambda_2 = \|p\|_2 = \beta^* \cdot \|\max(x - \alpha^* \cdot e,\ 0)\|_2$. We now show that the claimed representation is unique, and then present the two different proofs for the existence of the representation.

Uniqueness: It is $d \ge 2$, since $d = 0$ would violate that $\|p\|_1 > 0$ and $d = 1$ is impossible because $\sigma^* \ne 1$. We first show that there are two distinct indices $i, j \in I$ with $p_i \ne p_j$. Assume this was not the case, then $p_i =: \gamma$, say, for all $i \in I$. Let $j := \arg\min_{i\in I} x_i$ be the index of the smallest coordinate of x which has its index in I. Let $\psi := \left(\lambda_1 - \sqrt{d\lambda_2^2-\lambda_1^2}/\sqrt{d-1}\right)/d \in \mathbb{R}$ and $\omega := \lambda_1 - (d-1)\psi \in \mathbb{R}$ be numbers and define $s := \sum_{i\in I\backslash\{j\}} \psi e_i + \omega e_j \in \mathbb{R}^n$. Then $s \in T$ like in the argument where we have shown that T is non-empty. Because of $\|p\|_1 = \|s\|_1$ and $\|p\|_2 = \|s\|_2$, it follows that $\|x - p\|_2^2 - \|x - s\|_2^2 = 2x^T(s - p) = 2\sum_{i\in I\backslash\{j\}} x_i(\psi - \gamma) + 2x_j(\omega - \gamma) \ge 2x_j((d-1)\psi + \omega - d\gamma) = 2x_j\left(\|s\|_1 - \|p\|_1\right) = 0$. Hence s is at least as good an approximation to x as p, violating the uniqueness of p. Therefore, it is impossible that the set $\{p_i | i \in I\}$ is a singleton.

Now let $i, j \in I$ with $p_i \ne p_j$ and $\alpha_1^*, \alpha_2^*, \beta_1^*, \beta_2^* \in \mathbb{R}$ such that $p = \beta_1^* \cdot \max\left(x - \alpha_1^* \cdot e,\ 0\right) = \beta_2^* \cdot \max\left(x - \alpha_2^* \cdot e,\ 0\right)$. Clearly $\beta_1^* \ne 0$ and $\beta_2^* \ne 0$ as $d \ne 0$. It is $0 \ne p_i - p_j = \beta_1^*(x_i - \alpha_1^*) - \beta_1^*(x_j - \alpha_1^*) = \beta_1^*(x_i - x_j)$, thus $x_i \ne x_j$ holds. Moreover, $0 = p_i - p_j + p_j - p_i = (\beta_1^* - \beta_2^*)(x_i - x_j)$, hence $\beta_1^* = \beta_2^*$. Finally, we have that $0 = p_i - p_i = \beta_1^*(x_i - \alpha_1^*) - \beta_2^*(x_i - \alpha_2^*) = \beta_1^*(\alpha_2^* - \alpha_1^*)$, which yields $\alpha_1^* = \alpha_2^*$, and hence the representation is unique.

Existence (constructive): Let $H := \{a \in \mathbb{R}^n | e^T a = \lambda_1\}$ be the hyperplane on which all points in the non-negative orthant have an L_1 norm of λ_1 and let $C := \mathbb{R}^n_{\ge 0} \cap H$ be a scaled canonical simplex. Further, let $L := \{q \in H \,|\, \|q\|_2 = \lambda_2\}$ be a circle on H, and for an arbitrary index set $I \subseteq \{1,\dots,n\}$ let $L_I := \{a \in L | a_i = 0 \text{ for all } i \notin I\}$ be a subset of L where all coordinates not indexed by I vanish. With Theorem 2 and Appendix D from Thom and Palm (2013) there exists a finite sequence of index sets $I_1,\dots,I_h \subseteq \{1,\dots,n\}$ with $I_j \supsetneq I_{j+1}$ for $j \in \{1,\dots,h-1\}$ such that $\operatorname{proj}_T(x)$ is the result of the finite sequence

$$\begin{aligned}
r(0) &:= \operatorname{proj}_H(x), & s(0) &:= \operatorname{proj}_L(r(0)),\\
r(1) &:= \operatorname{proj}_C(s(0)), & s(1) &:= \operatorname{proj}_{L_{I_1}}(r(1)),\ \dots\\
r(j) &:= \operatorname{proj}_C(s(j-1)), & s(j) &:= \operatorname{proj}_{L_{I_j}}(r(j)),\ \dots\\
r(h) &:= \operatorname{proj}_C(s(h-1)), & s(h) &:= \operatorname{proj}_{L_{I_h}}(r(h)) = p.
\end{aligned}$$

All intermediate projections yield unique results because p was restricted to be unique. The index sets contain the indices of the entries that survive the projections onto C, $I_j := \{i \in \{1,\dots,n\} | r_i(j) \ne 0\}$ for $j \in \{1,\dots,h\}$. In other words, p can be computed from x by alternating projections, where the sets L and L_{I_j} are non-convex for all $j \in \{1,\dots,h\}$. The expressions for the individual projections are given in Lemma 13, Lemma 17, Proposition 24, and Lemma 30, respectively, in Thom and Palm (2013).

Let $I_0 := \{1,\dots,n\}$ for completeness, then we can define the following constants for $j \in \{0,\dots,h\}$. Let $d_j := |I_j|$ be the number of relevant coordinates in each iteration, and let

$$\beta_j := \sqrt{\frac{d_j\lambda_2^2 - \lambda_1^2}{d_j\|V_{I_j}x\|_2^2 - \|V_{I_j}x\|_1^2}} \text{ and } \alpha_j := \frac{1}{d_j}\left(\|V_{I_j}x\|_1 - \frac{\lambda_1}{\beta_j}\right)$$

be real numbers. We have that $d_j\lambda_2^2 - \lambda_1^2 \ge d_h\lambda_2^2 - \lambda_1^2 \ge 0$ by construction which implies $\beta_j > 0$ for all $j \in \{0,\dots,h\}$. We now claim that the following holds:

(a) $s_i(j) = \beta_j \cdot (x_i - \alpha_j)$ for all $i \in I_j$ for all $j \in \{0,\dots,h\}$.
(b) $\alpha_0 \le \dots \le \alpha_h$ and $\beta_0 \le \dots \le \beta_h$.
(c) $x_i \le \alpha_j$ for all $i \notin I_j$ for all $j \in \{0,\dots,h\}$.
(d) $p = \beta_h \cdot \max(x - \alpha_h \cdot e,\ 0)$.

We start by showing (a) with induction. For $j = 0$, we have $r(0) = x + {}^1/_n \cdot (\lambda_1 - \|x\|_1)\,e$ using Lemma 13 from Thom and Palm (2013). With Lemma 17 stated in Thom and Palm (2013), we have $s(0) = \delta r(0) + (1 - \delta)m$ with $m = {}^{\lambda_1}/_n \cdot e$ and $\delta^2 = \left(\lambda_2^2 - {}^{\lambda_1^2}/_n\right) / \|r(0) - m\|_2^2$. We see that

$\|r(0)-m\|_2^2 = \|x-\|x\|_1/n\cdot e\|_2^2 = \|x\|_2^2-\|x\|_1^2/n$ and therefore $\delta = \beta_0$, and thus $s(0) = \beta_0 \cdot (x - 1/n \cdot (\|x\|_1 - \lambda_1/\beta_0)\, e)$, so the claim holds for the base case.

Suppose that (a) holds for j and we want to show it also holds for $j+1$. It is $r(j+1) = \operatorname{proj}_C(s(j))$ by definition, and Proposition 31 in Thom and Palm (2013) implies $r(j+1) = \max\left(s(j) - \hat{t}\cdot e,\ 0\right)$ where $\hat{t} \in \mathbb{R}$ can be expressed explicitly as $\hat{t} = 1/d_{j+1} \cdot \left(\sum_{i\in I_{j+1}} s_i(j) - \lambda_1\right)$, which is the mean value of the entries that survive the simplex projection up to an additive constant. We note that $\hat{t}$ is here always non-negative, see Lemma 28(a) in Thom and Palm (2013), which we will need to show (b). Since $I_{j+1} \subsetneq I_j$ we yield $s_i(j) = \beta_j \cdot (x_i - \alpha_j)$ for all $i \in I_{j+1}$ with the induction hypothesis, and therefore we have that $\hat{t} = 1/d_{j+1} \cdot \left(\beta_j \|V_{I_{j+1}} x\|_1 - d_{j+1}\beta_j\alpha_j - \lambda_1\right)$. We find that $r_i(j+1) > 0$ for $i \in I_{j+1}$ by definition, and we can omit the rectifier so that $r_i(j+1) = s_i(j) - \hat{t}$. Using the induction hypothesis and the expression for $\hat{t}$ we have $r_i(j+1) = \beta_j x_i - \beta_j/d_{j+1} \cdot \|V_{I_{j+1}}x\|_1 + \lambda_1/d_{j+1}$. For projecting onto $L_{I_{j+1}}$, the distance between $r(j+1)$ and $m_{I_{j+1}} = \lambda_1/d_{j+1} \cdot \sum_{i\in I_{j+1}} e_i$ is required for computation of $\delta^2 = \left(\lambda_2^2 - \lambda_1^2/d_{j+1}\right) / \|r(j+1) - m_{I_{j+1}}\|_2^2$, so that Lemma 30 from Thom and Palm (2013) can be applied. We have that $\|r(j+1) - m_{I_{j+1}}\|_2^2 = \sum_{i\in I_{j+1}} \left(\beta_j x_i - \beta_j/d_{j+1} \cdot \|V_{I_{j+1}}x\|_1\right)^2 = \beta_j^2 \cdot \left(\|V_{I_{j+1}}x\|_2^2 - \|V_{I_{j+1}}x\|_1^2/d_{j+1}\right)$, and further $\delta = \beta_{j+1}/\beta_j$. Now let $i \in I_{j+1}$ be an index, then we have $s_i(j+1) = \delta r_i(j+1) + (1-\delta)\cdot \lambda_1/d_{j+1} = \beta_{j+1} \cdot \left(x_i - 1/d_{j+1} \cdot \left(\|V_{I_{j+1}}x\|_1 - \lambda_1/\beta_{j+1}\right)\right)$ using Lemma 30 from Thom and Palm (2013). Therefore (a) holds for all $j \in \{0, \ldots, h\}$.

Let us now turn to (b). From the last paragraph, we know that $\delta = \beta_{j+1}/\beta_j$ for all $j \in \{0, \ldots, h-1\}$ for the projections onto $L_{I_{j+1}}$. On the other hand, we have that $\|r(j+1) - m_{I_{j+1}}\|_2^2 = \|r(j+1)\|_2^2 - \lambda_1^2/d_{j+1}$ from the proof of Lemma 30(a) from Thom and Palm (2013), and $\|r(j+1)\|_2^2 \le \lambda_2^2$ holds from the proof of Lemma 28(f) in Thom and Palm (2013), so $\delta \ge 1$ which implies $\beta_0 \le \cdots \le \beta_h$. As noted above, the separator for projecting onto C satisfies $\hat{t} \ge 0$ for all $j \in \{0, \ldots, h-1\}$. By rearranging this inequality and using $\beta_j \le \beta_{j+1}$, we conclude that $\alpha_j \le 1/d_{j+1} \cdot \left(\|V_{I_{j+1}}x\|_1 - \lambda_1/\beta_j\right) \le \alpha_{j+1}$, hence $\alpha_0 \le \cdots \le \alpha_h$.

For (c) we want to show that the coordinates in the original vector x which will vanish in some iteration when projecting onto C are already small. The base case $j = 0$ of an induction for j is trivial since the complement of I_0 is empty. In the induction step, we note that the complement of I_{j+1} can be partitioned into $I_{j+1}^C = I_j^C \cup \left(I_j \cap I_{j+1}^C\right)$ since $I_{j+1} \subsetneq I_j$. For $i \in I_j^C$ we already know that $x_i \le \alpha_j \le \alpha_{j+1}$ by the induction hypothesis and (b). We have shown in (a) that $s_i(j) = \beta_j \cdot (x_i - \alpha_j)$ for all $i \in I_j$, and if $i \in I_j \backslash I_{j+1}$ then $s_i(j) \le \hat{t}$ since $0 = r_i(j+1) = \max\left(s_i(j) - \hat{t},\ 0\right)$. By substituting the explicit expression for $\hat{t}$ and solving for x_i we yield $x_i \le 1/d_{j+1} \cdot \left(\|V_{I_{j+1}}x\|_1 - \lambda_1/\beta_j\right) \le \alpha_{j+1}$, and hence the claim holds for all $i \notin I_{j+1}$.

If we can now show that (d) holds, then the claim of the theorem follows by setting $\alpha^* := \alpha_h$ and $\beta^* := \beta_h$. We note that by construction $p = s(h)$ and $s_i(h) \ge 0$ for all coordinates $i \in \{1, \ldots, n\}$. When $i \in I_h$, then $s_i(h) = \beta_h \cdot (x_i - \alpha_h)$ with (a), which is positive by requirement, so when the rectifier is applied nothing changes. If $i \notin I_h$ then $x_i - \alpha_h \le 0$ by (c), and indeed $\beta_h \cdot \max(x_i - \alpha_h,\ 0) = 0 = s_i(h)$. The expression therefore holds for all $i \in \{1, \ldots, n\}$, which completes the constructive proof of existence.

Existence (implicit): Existence of the projection is guaranteed by the Weierstraß extreme value theorem since T is compact. Now let $f\colon \mathbb{R}^n \to \mathbb{R}$, $s \mapsto \|s - x\|_2^2$, be the objective function, and let the constraints be represented by the functions $h_1\colon \mathbb{R}^n \to \mathbb{R}$, $s \mapsto e^T s - \lambda_1$, $h_2\colon \mathbb{R}^n \to \mathbb{R}$, $s \mapsto \|s\|_2^2 - \lambda_2^2$, and $g_i\colon \mathbb{R}^n \to \mathbb{R}$, $s \mapsto -s_i$, for all indices $i \in \{1, \ldots, n\}$. All these functions are continuously differentiable. If $p = \operatorname{proj}_T(x)$, then p is a local minimum of f subject to $h_1(p) = 0$, $h_2(p) = 0$ and $g_1(p) \le 0, \ldots, g_n(p) \le 0$.

For application of the method of Lagrange multipliers we first have to show that p is regular, which means that the gradients of h_1, h_2 and g_i for $i \notin I$ evaluated in p must be linearly independent (Bertsekas 1999, Section 3.3.1). Let $J := I^C = \{j_1, \ldots, j_{n-d}\}$, say, then $|J| \le n-2$ since $d \ge 2$. Hence we have at most n vectors from $\mathbb{R}^n$ for which we have to show linear independence. Clearly $h_1'(s) = e$, $h_2'(s) = 2s$ and $g_i'(s) = -e_i$ for all $i \in \{1, \ldots, n\}$. Now let $u_1, u_2 \in \mathbb{R}$ and $v \in \mathbb{R}^{n-d}$ with $u_1 e + 2u_2 p - \sum_{\mu=1}^{n-d} v_\mu e_{j_\mu} = 0 \in \mathbb{R}^n$. Then, let $\mu \in \{1, \ldots, n-d\}$, then $p_{j_\mu} = 0$ by definition of I and hence by pre-multiplication of the equation above with $e_{j_\mu}^T$ we yield $u_1 - v_\mu = 0 \in \mathbb{R}$. Therefore $u_1 = v_\mu$ for all $\mu \in \{1, \ldots, n-d\}$. On the other hand, if $i \in I$ then $p_i > 0$ and $e_i^T e_{j_\mu} = 0$ for all $\mu \in \{1, \ldots, n-d\}$. Hence $u_1 + 2u_2 p_i = 0 \in \mathbb{R}$ for all $i \in I$. In the first paragraph of the uniqueness proof we have shown there are two distinct indices $i, j \in I$ with $p_i \ne p_j$. Since $u_1 + 2u_2 p_i = 0 = u_1 + 2u_2 p_j$ and thus $0 = 2u_2(p_i - p_j)$ we can conclude that $u_2 = 0$, which implies $u_1 = 0$. Then $v_1 = \cdots = v_{n-d} = 0$ as well, which shows that p is regular.

The Lagrangian is $\mathscr{L}\colon \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}$, $(s,\ \alpha,\ \beta,\ \gamma) \mapsto f(s) + \alpha h_1(s) + \beta h_2(s) + \sum_{i=1}^n \gamma_i g_i(s)$, and its derivative with respect to its first argument s is given by $\mathscr{L}'(s,\ \alpha,\ \beta,\ \gamma) := \partial/\partial s\ \mathscr{L}(s,\ \alpha,\ \beta,\ \gamma) = 2(s-x) + \alpha\cdot e + 2\beta\cdot s - \gamma \in \mathbb{R}^n$. Now, Proposition 3.3.1 from Bertsekas (1999) guarantees the existence of Lagrange multipliers $\tilde{\alpha}, \tilde{\beta} \in \mathbb{R}$ and $\tilde{\gamma} \in \mathbb{R}^n$ with $\mathscr{L}'(p,\ \tilde{\alpha},\ \tilde{\beta},\ \tilde{\gamma}) = 0$, $\tilde{\gamma}_i \ge 0$ for all $i \in \{1, \ldots, n\}$ and $\tilde{\gamma}_i = 0$ for $i \in I$. Assume $\tilde{\beta} = -1$, then $2x = \tilde{\alpha}\cdot e - \tilde{\gamma}$ since the derivative of $\mathscr{L}$ must vanish. Hence $x_i = \tilde{\alpha}/2$ for all $i \in I$, and therefore $\{p_i \mid i \in I\}$ is a singleton with Remark 10 from Thom and Palm (2013) as p was assumed unique and T is permutation-invariant.

We have seen earlier that this is absurd, so $\tilde{\beta} \neq -1$ must hold.

Write $\alpha^* := \tilde{\alpha}/2$, $\beta^* := 1/(\tilde{\beta}+1)$ and $\gamma^* := \tilde{\gamma}/2$ for notational convenience. We then obtain $p = \beta^*(x - \alpha^* \cdot e + \gamma^*)$ because $\mathscr{L}'$ vanishes. Then $h_1(p) = 0$ implies that $\lambda_1 = \sum_{i \in I} p_i = \sum_{i \in I} \beta^*(x_i - \alpha^*) = \beta^*(\|V_I x\|_1 - d\alpha^*)$, and with $h_2(p) = 0$ follows that $\lambda_2^2 = \sum_{i \in I} p_i^2 = (\beta^*)^2 \cdot (\|V_I x\|_2^2 - 2\alpha^* \|V_I x\|_1 + d \cdot (\alpha^*)^2)$. By taking the ratio λ_1^2/λ_2^2 and after elementary algebraic transformations we arrive at the quadratic equation $a \cdot (\alpha^*)^2 + b \cdot \alpha^* + c = 0$, where $a := d \cdot \left(d - \lambda_1^2/\lambda_2^2\right)$, $b := 2\|V_I x\|_1 \cdot \left(\lambda_1^2/\lambda_2^2 - d\right)$ and $c := \|V_I x\|_1^2 - \|V_I x\|_2^2 \cdot \lambda_1^2/\lambda_2^2$ are reals. The discriminant is $\Delta := b^2 - 4ac = 4 \cdot \lambda_1^2/\lambda_2^2 \cdot \left(d - \lambda_1^2/\lambda_2^2\right) \cdot \left(d\|V_I x\|_2^2 - \|V_I x\|_1^2\right)$. Since $V_I x \in \mathbb{R}^d$ we have that $d\|V_I x\|_2^2 - \|V_I x\|_1^2 \geq 0$. Moreover, the number d is not arbitrary. As p exists by the Weierstraß extreme value theorem with $\|p\|_0 = d$, $\|p\|_1 = \lambda_1$ and $\|p\|_2 = \lambda_2$, we find that $\lambda_1 \leq \sqrt{d}\lambda_2$ and hence $\Delta \geq 0$, so there must be a real solution to the equation above. Solving the equation shows that

$$\alpha^* \in \left\{ \frac{1}{d}\left(\|V_I x\|_1 \pm \lambda_1 \sqrt{\frac{d\|V_I x\|_2^2 - \|V_I x\|_1^2}{d\lambda_2^2 - \lambda_1^2}} \right) \right\},$$

hence from $h_1(p) = 0$ we obtain

$$\beta^* \in \left\{ \mp\sqrt{d\lambda_2^2 - \lambda_1^2} \Big/ \sqrt{d\|V_I x\|_2^2 - \|V_I x\|_1^2} \right\}.$$

Suppose α^* is the number that arises from the "+" before the square root, then β^* is the number with the "–" sign, thus $\beta^* < 0$. We have seen earlier that there are two distinct indices $i, j \in I$ with $p_i \neq p_j$. We can assume $p_i > p_j$, then $0 < p_i - p_j = \beta^*(x_i - x_j)$ which implies that $x_i < x_j$. This is not possible as it violates the order-preservation property of projections onto permutation-invariant sets (Lemma 9(a) from Thom and Palm 2013). Thus our choice of α^* was not correct in the first place, and α^* must be as stated in the claim.

It remains to be shown that p is the result of a soft-shrinkage function. If $i \in I$, then $0 < p_i = \beta^*(x_i - \alpha^*)$, and $\beta^* > 0$ shows $x_i > \alpha^*$ such that $p_i = \beta^* \cdot \max(x_i - \alpha^*,\ 0)$. When $i \notin I$, we have $0 = p_i = \beta^*(x_i - \alpha^* + \gamma_i^*)$ where $\gamma_i^* \geq 0$ and still $\beta^* > 0$, thus $x_i \leq \alpha^*$ and $p_i = \beta^* \cdot \max(x_i - \alpha^*,\ 0)$ holds. Therefore, the representation holds for all entries. □

Finding the set I containing the indices of the positive coordinates of the projection result is the key for algorithmic computation of the projection. Based on the constructive proof this could, for example, be achieved by carrying out alternating projections whose run-time complexity is between quasi-linear and quadratic in the problem dimensionality n and whose space complexity is linear. An alternative is the method proposed by Potluru et al. (2013), where the input vector is sorted and then each possible candidate for I is checked. Due to the sorting, I must be of the form $I = \{1, \ldots, d\}$, where now only d is unknown (see also the proof of Theorem 3 from Thom and Palm 2013). Here, the run-time complexity is quasi-linear and the space complexity is linear in the problem dimensionality since also the sorting permutation has to be stored. When n gets large, algorithms with a smaller computational complexity are mandatory.

Properties of the Auxiliary Function

We have already informally introduced the auxiliary function in Sect. 2.2. Here is a satisfactory definition:

Definition 2 Let $x \in \mathbb{R}^n_{\geq 0} \setminus T$ be a point such that $\operatorname{proj}_T(x)$ is unique and $\sigma(x) < \sigma^*$. Let $x_{\max}$ denote the maximum entry of x, then

$$\Psi\colon\ [0,\ x_{\max}) \to \mathbb{R}, \qquad \alpha \mapsto \frac{\|\max(x - \alpha \cdot e,\ 0)\|_1}{\|\max(x - \alpha \cdot e,\ 0)\|_2} - \frac{\lambda_1}{\lambda_2}$$

is called *auxiliary function* for the projection onto T.

We call Ψ *well-defined* if all requirements from the definition are met. Note that the situation where $\sigma(x) \geq \sigma^*$ is trivial, because in this sparseness-decreasing setup we have that all coordinates of the projection must be positive. Hence $I = \{1, \ldots, n\}$ in Theorem 1, and α^* can be computed with the there provided formula.

We need more notation to describe the properties of Ψ. Let $x \in \mathbb{R}^n$ be a point. We write $\mathscr{X} := \{x_i \mid i \in \{1, \ldots, n\}\}$ for the set that contains the entries of x. Let $x_{\min} := \min \mathscr{X}$ be short for the smallest entry of x, and $x_{\max} := \max \mathscr{X}$ and $x_{\text{2nd-max}} := \max \mathscr{X} \setminus \{x_{\max}\}$ denote the two largest entries of x. Further, $q\colon \mathbb{R} \to \mathbb{R}^n$, $\alpha \mapsto \max(x - \alpha \cdot e,\ 0)$, denotes the curve that evolves from application of the soft-shrinkage function to x. The Manhattan norm and Euclidean norm of points from q is given by $\ell_1\colon \mathbb{R} \to \mathbb{R}$, $\alpha \mapsto \|q(\alpha)\|_1$, and $\ell_2\colon \mathbb{R} \to \mathbb{R}$, $\alpha \mapsto \|q(\alpha)\|_2$, respectively. Therefore, $\Psi = \ell_1/\ell_2 - \lambda_1/\lambda_2$ and $\tilde{\Psi}$ from Sect. 4.1 can be written as $\ell_1^2/\ell_2^2 - \lambda_1^2/\lambda_2^2$, such that its derivative can be expressed in terms of Ψ' using the chain rule.

The next result provides statements on the auxiliary function's analytical nature and links its zero with the projection onto T:

Lemma 3 *Let $x \in \mathbb{R}^n_{\geq 0} \setminus T$ be given such that the auxiliary function Ψ is well-defined. Then the following holds:*

(a) Ψ is continuous on $[0,\ x_{\max})$.
(b) Ψ is differentiable on $[0,\ x_{\max}) \setminus \mathscr{X}$.
(c) Ψ is strictly decreasing on $[0,\ x_{\text{2nd-max}})$, and on $[x_{\text{2nd-max}},\ x_{\max})$ it is constant.
(d) There is exactly one $\alpha^* \in (0,\ x_{\text{2nd-max}})$ with $\Psi(\alpha^*) = 0$. It is then $\operatorname{proj}_T(x) = (\lambda_2/\ell_2(\alpha^*)) \cdot q(\alpha^*)$.

Proof In addition to the original claim, we also give explicit expressions for the derivative of Ψ and higher derivatives thereof in part (c). These are necessary to show that Ψ is strictly decreasing and constant, respectively, on the claimed intervals and for the explicit implementation of Algorithm 2.

(a) The function q is continuous because so is the soft-shrinkage function. Hence ℓ_1, ℓ_2 and Ψ are continuous as compositions of continuous functions.

(b) The soft-shrinkage function is differentiable everywhere except at its offset. Therefore, Ψ is differentiable everywhere except for when its argument coincides with an entry of x, that is on $[0,\ x_{\max}) \setminus \mathscr{X}$.

(c) We start with deducing the first and second derivative of Ψ. Let $x_j, x_k \in \mathscr{X} \cup \{0\}$, $x_j < x_k$, such that there is no element from $\mathscr{X}$ between them. We here allow $x_j = 0$ and $x_k = x_{\min}$ when $0 \notin \mathscr{X}$ for completeness. Then the index set $I := \{i \in \{1, \ldots, n\} | x_i > \alpha\}$ of non-vanishing coordinates in q is constant for $\alpha \in \left(x_j,\ x_k\right)$, and the derivative of Ψ can be computed using a closed-form expression. For this, let $d := |I|$ denote the number of non-vanishing coordinates in q on that interval. With $\ell_1(\alpha) = \sum_{i \in I} (x_i - \alpha) = \sum_{i \in I} x_i - d\alpha$ we obtain $\ell_1'(\alpha) = -d$. Analogously, it is $\partial/\partial\alpha\ \ell_2(\alpha)^2 = \partial/\partial\alpha\ \sum_{i \in I} (x_i - \alpha)^2 = -2\ell_1(\alpha)$, and the chain rule yields $\ell_2'(\alpha) = \frac{\partial}{\partial\alpha}\sqrt{\ell_2(\alpha)^2} = -\ell_1(\alpha)/\ell_2(\alpha)$. Application of the quotient rule gives $\Psi'(\alpha) = \left(\ell_1(\alpha)^2/\ell_2(\alpha)^2 - d\right) / \ell_2(\alpha)$. The second derivative is of similar form. We find $\partial/\partial\alpha\ \ell_1(\alpha)^2 = -2d\ell_1(\alpha)$, and hence $\partial/\partial\alpha\ \left(\ell_1(\alpha)^2/\ell_2(\alpha)^2\right) = 2\left(\ell_1(\alpha)/\ell_2(\alpha)^2\right) \cdot \left(\ell_1(\alpha)^2/\ell_2(\alpha)^2 - d\right)$. We have $\partial/\partial\alpha\ \left(1/\ell_2(\alpha)\right) = \ell_1(\alpha)/\ell_2(\alpha)^3$ and with the product rule we see that $\Psi''(\alpha) = 3\left(\ell_1(\alpha)/\ell_2(\alpha)^3\right) \cdot \left(\ell_1(\alpha)^2/\ell_2(\alpha)^2 - d\right) = 3\Psi'(\alpha) \cdot \ell_1(\alpha)/\ell_2(\alpha)^2$.

First let $\alpha \in (x_{\text{2nd-max}},\ x_{\max})$. It is then $d = 1$, that is q has exactly one non-vanishing coordinate. In this situation we find $\ell_1(\alpha) = \ell_2(\alpha)$ and $\Psi' \equiv 0$ on $(x_{\text{2nd-max}},\ x_{\max})$, thus Ψ is constant on $(x_{\text{2nd-max}},\ x_{\max})$ as a consequence of the mean value theorem from real analysis. Because Ψ is continuous, it is constant even on $[x_{\text{2nd-max}},\ x_{\max})$.

Next let $\alpha \in [0,\ x_{\text{2nd-max}}) \setminus \mathscr{X}$, and let x_j, x_k, I and d as in the first paragraph. We have $d \geq 2$ since $\alpha < x_{\text{2nd-max}}$. It is furthermore $\ell_1(\alpha) \leq \sqrt{d}\ell_2(\alpha)$ as $d = \|q(\alpha)\|_0$. This inequality is in fact strict, because $q(\alpha)$ has at least two distinct nonzero entries. Hence Ψ' is negative on the interval $\left(x_j,\ x_k\right)$, and the mean value theorem guarantees that Ψ is strictly decreasing on this interval. This property holds for the entire interval $[0,\ x_{\text{2nd-max}})$ due to the continuity of Ψ.

(d) The requirement $\sigma(x) < \sigma^*$ implies $\|x\|_1/\|x\|_2 > \lambda_1/\lambda_2$ and thus $\Psi(0) > 0$. Let $\alpha \in (x_{\text{2nd-max}},\ x_{\max})$ be arbitrary, then $\ell_1(\alpha) = \ell_2(\alpha)$ as in (c), and hence $\Psi(\alpha) < 0$ since $\lambda_2 < \lambda_1$ must hold. The existence of $\alpha^* \in [0,\ x_{\text{2nd-max}})$ with $\Psi(\alpha^*) = 0$ then follows from the intermediate value theorem and (c). Uniqueness of α^* is guaranteed because Ψ is strictly monotone on the relevant interval.

Define $p := \operatorname{proj}_T(x)$, then with Theorem 1 there is an $\tilde{\alpha} \in \mathbb{R}$ so that $p = \left(\lambda_2/\|\max(x - \tilde{\alpha} \cdot e,\ 0)\|_2\right) \cdot \max\left(x - \tilde{\alpha} \cdot e,\ 0\right) = \left(\lambda_2/\ell_2(\tilde{\alpha})\right) \cdot q(\tilde{\alpha})$. Since $p \in T$ we obtain $\Psi(\tilde{\alpha}) = 0$, and the uniqueness of the zero of Ψ implies that $\alpha^* = \tilde{\alpha}$. □

As described in Sect. 2.3, the exact value of the zero α^* of Ψ can be found by inspecting the neighboring entries in x of a candidate offset α. Let $x_j, x_k \in \mathscr{X}$ be these neighbors with $x_j \leq \alpha < x_k$ such that there is no element from $\mathscr{X}$ between x_j and x_k. When Ψ changes its sign from x_j to x_k, we know that its root must be located within this interval. Further, we then know that all coordinates with a value greater than x_j must survive the sparseness projection, which yields the set I from Theorem 1 and thus the explicit representation of the projection. The next result gathers these ideas and shows that it is easy to verify whether a change of sign in Ψ is on hand.

Lemma 4 *Let $x \in \mathbb{R}^n_{\geq 0} \setminus T$ be given such that Ψ is well-defined and let $\alpha \in [0,\ x_{\max})$ be arbitrary. If $\alpha < x_{\min}$, let $x_j := 0$ and $x_k := x_{\min}$. Otherwise, let $x_j := \max\{x_i | x_i \in \mathscr{X}$ and $x_i \leq \alpha\}$ be the left neighbor and let $x_k := \min\{x_i | x_i \in \mathscr{X}$ and $x_i > \alpha\}$ be the right neighbor of α. Both exist as the sets where the maximum and the minimum are taken are nonempty. Define $I := \{i \in \{1, \ldots, n\} | x_i > \alpha\}$ and $d := |I|$. Then:*

(a) When $\Psi(x_j) \geq 0$ and $\Psi(x_k) < 0$ then there is exactly one number $\alpha^* \in \left[x_j,\ x_k\right)$ with $\Psi(\alpha^*) = 0$.
(b) It is $\ell_1(\xi) = \|V_I x\|_1 - d\xi$ and $\ell_2^2(\xi) = \|V_I x\|_2^2 - 2\xi\,\|V_I x\|_1 + d\xi^2$ for $\xi \in \{x_j,\ \alpha,\ x_k\}$.
(c) If the inequalities $\lambda_2\ell_1(x_j) \geq \lambda_1\ell_2(x_j)$ and $\lambda_2\ell_1(x_k) < \lambda_1\ell_2(x_k)$ are satisfied and $p := \operatorname{proj}_T(x)$ denotes the projection of x onto T, then $I = \{i \in \{1, \ldots, n\} | p_i > 0\}$ and hence p can be computed exactly with Theorem 1.

Proof The claim in (a) is obvious with Lemma 3.

(b) We find that $\ell_1(\alpha) = \sum_{i \in I}(x_i - \alpha) = \|V_I x\|_1 - d\alpha$ and $\ell_2(\alpha)^2 = \sum_{i \in I}(x_i - \alpha)^2 = \|V_I x\|_2^2 - 2\alpha\|V_I x\|_1 + d\alpha^2$.

We have $K = I \setminus \tilde{K}$ with $K := \{i \in \{1, \ldots, n\} | x_i > x_k\}$ and $\tilde{K} := \{i \in \{1, \ldots, n\} | x_i = x_k\}$. One yields that $\ell_1(x_k) = \sum_{i \in K}(x_i - x_k) = \sum_{i \in I}(x_i - x_k) - \sum_{i \in \tilde{K}}(x_i - x_k) = \|V_I x\|_1 - dx_k$. The claim for $\ell_2(x_k)^2$ follows analogously.

Likewise $I = J \setminus \tilde{J}$ with $J := \{i \in \{1, \ldots, n\} | x_i > x_j\}$ and $\tilde{J} := \{i \in \{1, \ldots, n\} | x_i = x_j\}$, hence follows $\ell_1(x_j) = \sum_{i \in J}(x_i - x_j) = \sum_{i \in I}(x_i - x_j) + \sum_{i \in \tilde{J}}(x_i - x_j) = \|V_I x\|_1 - dx_j$. The value of $\ell_2(x_j)^2$ can be computed in the same manner.

(c) The condition in the claim is equivalent to the case of $\Psi(x_j) \geq 0$ and $\Psi(x_k) < 0$, hence with (a) there is a number $\alpha^* \in \left[x_j,\ x_k\right)$ with $\Psi(\alpha^*) = 0$. Note that $\alpha \neq \alpha^*$ in general. Write $p := \operatorname{proj}_T(x)$ and let $J := \{i \in \{1, \ldots, n\} | p_i > 0\}$. With Theorem 1 follows that $i \in J$ if and only if $x_i > \alpha^*$.

But this is equivalent to $x_i > x_j$, which in turn is equivalent to $x_i > \alpha$, therefore $I = J$ must hold. Thus we already had the correct set of non-vanishing coordinates of the projection in the first place, and α^* and β^* can be computed exactly using the formula from the claim of Theorem 1, which yields the projection p. □

Proof of Correctness of Projection Algorithm

In Sect. 2.3, we informally described our proposed algorithm for carrying out sparseness-enforcing projections, and provided a simplified flowchart in Fig. 3. After the previous theoretical considerations, we now propose and prove the correctness of a formalized algorithm for the projection problem. Here, the overall method is split into an algorithm that evaluates the auxiliary function Ψ and, based on its derivative, returns additional information required for finding its zero (Algorithm 2). The other part, Algorithm 3, implements the root-finding procedure and carries out the necessary computations to yield the result of the projection. It furthermore returns information that will be required for computation of the projection's gradient, which we will discuss below.

Theorem 5 *Let $x \in \mathbb{R}^n_{\geq 0}$ and $p := \operatorname{proj}_T(x)$ be unique. Then Algorithm 3 computes p in a number of operations linear in the problem dimensionality n and with only constant additional space.*

Proof We start by analyzing Algorithm 2, which evaluates Ψ at any given position α. The output includes the values of the auxiliary function, its first and second derivative, and the value of the transformed auxiliary function and its derivative. There is moreover a Boolean value which indicates whether the interval with the sign change of Ψ has been found, and three additional numbers required to compute the zero α^* of Ψ as soon as the correct interval has been found.

Let $I := \{i \in \{1, \dots, n\} \mid x_i > \alpha\}$ denote the indices of all entries in x larger than α. In the blocks from Line 1 to Line 11, the algorithm scans through all the coordinates of x. It identifies the elements of I, and computes numbers ℓ_1, ℓ_2^2, d, x_j and x_k on the fly. After Line 11, we clearly have that $\ell_1 = \|V_I x\|_1$, $\ell_2^2 = \|V_I x\|_2^2$ and $d = |I|$. Additionally, x_j and x_k are the left and right neighbors, respectively, of α. Therefore, the requirements of Lemma 4 are satisfied.

The next two blocks spanning from Line 12 to Line 17 compute scalar numbers according to Lemma 4(b), the definition of Ψ, the first two derivatives thereof given explicitly in the proof of Lemma 3(c), the definition of $\tilde{\Psi}$ and its derivative given by the chain rule. In Line 18, it is checked whether

Algorithm 2: Linear time and constant space evaluation of the auxiliary function Ψ

```
Input: Point to be projected $x \in \mathbb{R}^n_{\geq 0}$, target norms $\lambda_1, \lambda_2 \in \mathbb{R}$,
       position $\alpha \in [0,\ x_{\max})$ where $\Psi$ should be evaluated.
Output: Values $\Psi(\alpha),\ \Psi'(\alpha),\ \Psi''(\alpha),\ \tilde{\Psi}(\alpha),\ \tilde{\Psi}'(\alpha) \in \mathbb{R}$,
        finished $\in$ {true, false} indicating whether the correct
        interval has been found, numbers $\ell_1, \ell_2^2 \in \mathbb{R}$ and $d \in \mathbb{N}$
        needed to exactly compute $\alpha^*$ with $\Psi(\alpha^*) = 0$.
    // Initialize.
1   $\ell_1 := 0;\ \ell_2^2 := 0;\ d := 0;$
2   $x_j := 0;\ \Delta x_j := -\alpha;\ x_k := \infty;\ \Delta x_k := \infty;$
    // Scan through $x$.
3   for $i := 1$ to $n$ do
4       $t := x_i - \alpha;$
5       if $t > 0$ then
6           $\ell_1 := \ell_1 + x_i;\ \ell_2^2 := \ell_2^2 + x_i^2;\ d := d + 1;$
7           if $t < \Delta x_k$ then $x_k := x_i;\ \Delta x_k := t$ ;
8       else
9           if $t > \Delta x_j$ then $x_j := x_i;\ \Delta x_j := t$ ;
10      end
11  end
    // Compute $\Psi(\alpha)$, $\Psi'(\alpha)$ and $\Psi''(\alpha)$.
12  $\ell_1(\alpha) := \ell_1 - d\alpha;\ \ell_2(\alpha)^2 := \ell_2^2 - 2\alpha\ell_1 + d\alpha^2;$
13  $\Psi(\alpha) := \ell_1(\alpha)/\sqrt{\ell_2(\alpha)^2} - \lambda_1/\lambda_2;$
14  $\Psi'(\alpha) := \left(\ell_1(\alpha)^2/\ell_2(\alpha)^2 - d\right)/\sqrt{\ell_2(\alpha)^2};$
15  $\Psi''(\alpha) := 3\Psi'(\alpha) \cdot \ell_1(\alpha)/\ell_2(\alpha)^2;$
    // Compute $\tilde{\Psi}(\alpha)$ and $\tilde{\Psi}'(\alpha)$.
16  $\tilde{\Psi}(\alpha) := \ell_1(\alpha)^2/\ell_2(\alpha)^2 - \lambda_1^2/\lambda_2^2;$
17  $\tilde{\Psi}'(\alpha) := 2\ell_1(\alpha)/\sqrt{\ell_2(\alpha)^2} \cdot \Psi'(\alpha);$
    // Check for sign change from $\Psi(x_j)$ to
       $\Psi(x_k)$.
18  finished $:= \lambda_2(\ell_1 - dx_j) \geq \lambda_1\sqrt{\ell_2^2 - 2x_j\ell_1 + dx_j^2}$ and
                $\lambda_2\,(\ell_1 - dx_k) < \lambda_1\sqrt{\ell_2^2 - 2x_k\ell_1 + dx_k^2};$
19  return $\left(\Psi(\alpha), \Psi'(\alpha), \Psi''(\alpha), \tilde{\Psi}(\alpha), \tilde{\Psi}'(\alpha), \text{finished}, \ell_1, \ell_2^2, d\right);$
```

the conditions from Lemma 4(c) hold using the statements from Lemma 4(b). The result is stored in the Boolean variable "finished".

Finally all computed numbers are passed back for further processing. Algorithm 2 clearly needs time linear in n and only constant additional space.

Algorithm 3 performs the actual projection in-place, and outputs values needed for the gradient of the projection. It uses Algorithm 2 as sub-program by calls to the function "auxiliary". The algorithm first checks whether $\Psi(0) \leq 0$, which is fulfilled when $\sigma(x) \geq \sigma^*$. In this case, all coordinates survive the projection and computation of α^* is straightforward with Theorem 1 using $I = \{1, \dots, n\}$.

Otherwise, Lemma 3(d) states that $\alpha^* \in (0,\ x_{\text{2nd-max}})$. We can find α^* numerically with standard root-finding algorithms since Ψ is continuous and strictly decreasing on the interval $(0,\ x_{\text{2nd-max}})$. The concrete variant is chosen

Algorithm 3: Linear time and constant space algorithm for projections onto T. The auxiliary function Ψ is evaluated by calls to "auxiliary", which are carried out by Algorithm 2. This algorithm operates in-place, the input vector is overwritten by the output vector upon completion

```
Input: Point to be projected $x \in \mathbb{R}^n_{\geq 0}$, target norms $\lambda_1, \lambda_2 \in \mathbb{R}$,
       solver $\in$ {Bisection, Newton, NewtonSqr, Halley}.
Output: Projection $\text{proj}_T(x) \in T$ as first element replacing the
        input vector, numbers $\ell_1, \ell_2^2 \in \mathbb{R}$ and $d \in \mathbb{N}$ needed to
        compute the projection's gradient with Algorithm 4.
// Skip root-finding if decreasing
   sparseness.
1  $\left(\Psi(\alpha), \Psi'(\alpha), \Psi''(\alpha), \tilde{\Psi}(\alpha), \tilde{\Psi}'(\alpha), \text{finished}, \ell_1, \ell_2^2, d\right) :=$
   auxiliary($x$, $\lambda_1$, $\lambda_2$, 0);
2  if $\Psi(\alpha) \leq 0$ then go to Line 19;
   // Sparseness should be increased.
3  lo := 0; up := $x_{\text{2nd-max}}$; $\alpha :=$ lo $+ 1/2 \cdot$ (up − lo);
4  $\left(\Psi(\alpha), \Psi'(\alpha), \Psi''(\alpha), \tilde{\Psi}(\alpha), \tilde{\Psi}'(\alpha), \text{finished}, \ell_1, \ell_2^2, d\right) :=$
   auxiliary($x$, $\lambda_1$, $\lambda_2$, $\alpha$);
   // Start root-finding procedure.
5  while not finished do
       // Update bisection interval.
6      if $\Psi(a) > 0$ then lo := $\alpha$ else up := $\alpha$;
       // One iteration of root-finding.
7      if solver = Bisection then $\alpha :=$ lo $+ 1/2 \cdot$ (up − lo);
8      else // Use solvers based on derivatives.
9          if solver = Newton then $\alpha := \alpha - \Psi(\alpha)/\Psi'(\alpha)$;
10         else if solver = NewtonSqr then $\alpha := \alpha - \tilde{\Psi}(\alpha)/\tilde{\Psi}'(\alpha)$;
11         else if solver = Halley then
12             $h := 1 - (\Psi(\alpha)\Psi''(\alpha))/(2\Psi'(\alpha)^2)$;
13             $\alpha := \alpha - \Psi(\alpha)/\left(\max(0.5, \min(1.5, h)) \cdot \Psi'(\alpha)\right)$;
14         end
           // Use bisection if $\alpha$ out of bounds.
15         if $\alpha <$ lo or $\alpha >$ up then $\alpha :=$ lo $+ 1/2 \cdot$ (up − lo);
16     end
       // Evaluate auxiliary function anew.
17     $\left(\Psi(\alpha), \Psi'(\alpha), \Psi''(\alpha), \tilde{\Psi}(\alpha), \tilde{\Psi}'(\alpha), \text{finished}, \ell_1, \ell_2^2, d\right) :=$
       auxiliary($x$, $\lambda_1$, $\lambda_2$, $\alpha$);
18 end
   // Correct interval has been found, compute
      $\alpha^*$ using closed-form expression.
19 $\alpha^* := \frac{1}{d}\left(\ell_1 - \lambda_1\sqrt{d\ell_2^2 - \ell_1^2}\Big/\sqrt{d\lambda_2^2 - \lambda_1^2}\right)$;
   // Compute result of the projection
      in-place.
20 $\rho := 0$;
21 for $i := 1$ to $n$ do
22     $t := x_i - \alpha^*$; if $t > 0$ then $x_i := t$; $\rho := \rho + t^2$ else $x_i := 0$;
23 end
24 for $i := 1$ to $n$ do $x_i := (\lambda_2/\sqrt{\rho}) \cdot x_i$;
25 return $(x, \ell_1, \ell_2^2, d)$;
```

by the parameter "solver" of Algorithm 3, implementation details can be found in Traub (1964) and Press et al. (2007).

Here, the root-finding loop starting at Line 5 is terminated once Algorithm 2 indicates that the correct interval for exact computation of the zero α^* has been identified. It is therefore not necessary to carry out root-finding until numerical convergence, it is enough to only come sufficiently close to α^*. Line 19 computes α^* based on the projection representation given in Theorem 1. This line is either reached directly from Line 2 if $\sigma(x) \geq \sigma^*$, or when the statements from Lemma 4(c) hold. The block starting at Line 20 computes $\max(x - \alpha^* \cdot e,\ 0)$ and stores this point's squared Euclidean norm in the variable ρ. Line 24 computes the number β^* from Theorem 1 and multiplies every entry of $\max(x - \alpha^* \cdot e,\ 0)$ with it, such that x finally contains the projection onto T. It would also be possible to create a new vector for the projection result and leave the input vector untouched, at the expense of additional memory requirements which are linear in the problem dimensionality.

When solver = Bisection, the loop in Line 5 is repeated a constant number of times regardless of n (see Sect. 2.3), and since Algorithm 2 terminates in time linear in n, Algorithm 3 needs time only linear in n. Further, the amount of additional memory needed is independent of n, as for Algorithm 2, such that the overall space requirements are constant. Therefore, Algorithm 3 is asymptotically optimal in the sense of complexity theory. □

Gradient of the Projection

Thom and Palm (2013) have shown that the projection onto T can be grasped as a function almost everywhere which is differentiable almost everywhere. An explicit expression for the projection's gradient was derived, which depended on the number of alternating projections required for carrying out the projection. Based on the characterization we gained through Theorem 1, we can derive a much simpler expression for the gradient which is also more efficient to compute:

Theorem 6 *Let $x \in \mathbb{R}^n_{\geq 0} \backslash T$ such that $p := \text{proj}_T(x)$ is unique. Let $\alpha^*, \beta^* \in \mathbb{R}$, $I \subseteq \{1, \ldots, n\}$ and $d := |I|$ be given as in Theorem 1. When $x_i \neq \alpha^*$ for all $i \in \{1, \ldots, n\}$, then proj_T is differentiable in x with $\partial/\partial x\ \text{proj}_T(x) = V_I^T G V_I$, where the matrix $G \in \mathbb{R}^{d \times d}$ is given by*

$$G := \sqrt{\tfrac{b}{a}}E_d - \tfrac{1}{\sqrt{ab}}\left(\lambda_2^2\tilde{e}\tilde{e}^T + d\tilde{p}\tilde{p}^T - \lambda_1\left(\tilde{e}\tilde{p}^T + \tilde{p}\tilde{e}^T\right)\right),$$

with $a := d\,\|V_I x\|_2^2 - \|V_I x\|_1^2 \in \mathbb{R}_{\geq 0}$ and $b := d\lambda_2^2 - \lambda_1^2 \in \mathbb{R}_{\geq 0}$. Here, $E_d \in \mathbb{R}^{d \times d}$ is the identity matrix, $\tilde{e} := V_I e \in \{1\}^d$ is the point where all coordinates are unity, and $\tilde{p} := V_I p \in \mathbb{R}^d_{>0}$.

Proof When $x_i \neq \alpha^*$ for all $i \in \{1, \ldots, n\}$, then I and d are invariant to local changes in x. Therefore, α^*, β^* and proj_T

are differentiable as composition of differentiable functions. In the following, we derive the claimed expression of the gradient of proj_T in x.

Let $\tilde{x} := V_I x \in \mathbb{R}^d$, then $\alpha^* = {}^1\!/_d \cdot \left(\|\tilde{x}\|_1 - \lambda_1\sqrt{a/b}\right)$. Define $\tilde{q} := V_I \cdot \max\left(x - \alpha^* \cdot e,\ 0\right)$, then $\tilde{q} = \tilde{x} - \alpha^* \cdot \tilde{e} \in \mathbb{R}^d_{>0}$ because $x_i > \alpha^*$ for all $i \in I$. Further $\tilde{p} = \lambda_2 \cdot \tilde{q}/\|\tilde{q}\|_2$, and we have $p = V_I^T V_I p = V_I^T \tilde{p}$ since $p_i = 0$ for all $i \notin I$. Application of the chain rule yields

$$\frac{\partial p}{\partial x} = \frac{\partial V_I^T \tilde{p}}{\partial \tilde{p}} \cdot \frac{\partial}{\partial \tilde{q}}\left(\lambda_2 \cdot \frac{\tilde{q}}{\|\tilde{q}\|_2}\right) \cdot \frac{\partial\left(\tilde{x} - \alpha^* \cdot \tilde{e}\right)}{\partial \tilde{x}} \cdot \frac{\partial V_I x}{\partial x},$$

and with $H := \lambda_2 \cdot \left(\partial/\partial\tilde{q}\ \left(\tilde{q}/\|\tilde{q}\|_2\right)\right) \cdot \left(\partial/\partial\tilde{x}\ \left(\tilde{x} - \alpha^* \cdot \tilde{e}\right)\right) \in \mathbb{R}^{d \times d}$ follows $\partial p/\partial x = V_I^T H V_I$, thus it only remains to show $G = H$.

One obtains $\partial/\partial\tilde{q}\ \left(\tilde{q}/\|\tilde{q}\|_2\right) = \left(E_d - \tilde{q}\tilde{q}^T/\|\tilde{q}\|_2^2\right)/\|\tilde{q}\|_2$. Since all the entries of $\tilde{q}$ and $\tilde{x}$ are positive, their L_1 norms equal the dot product with $\tilde{e}$. We have $\|\tilde{q}\|_1 = \|V_I x\|_1 - d\alpha^* = \lambda_1\sqrt{a/b}$, and we obtain that $\|\tilde{q}\|_2^2 = \|\tilde{x}\|_2^2 - \alpha^* \cdot \left(\|\tilde{x}\|_1 + \lambda_1\sqrt{a/b}\right) = \|\tilde{x}\|_2^2 - {}^1\!/_d \cdot \left(\|\tilde{x}\|_1^2 - \lambda_1^2 \cdot a/b\right) = a/d \cdot \left(1 + \lambda_1^2/b\right) = \lambda_2^2 \cdot a/b$. Now we can compute the gradient of α. Clearly b is independent of $\tilde{x}$. It is $\partial a/\partial\tilde{x} = 2d\tilde{x}^T - 2\|\tilde{x}\|_1\tilde{e}^T \in \mathbb{R}^{1 \times d}$. Since $\tilde{x} = \tilde{q} + \alpha^* \cdot \tilde{e}$ follows $d\tilde{x} - \|\tilde{x}\|_1\tilde{e} = d\tilde{q} - \lambda_1\sqrt{a/b} \cdot \tilde{e}$, and hence $\partial\sqrt{a}/\partial\tilde{x} = \sqrt{1/a} \cdot \left(d\tilde{q} - \lambda_1\sqrt{a/b} \cdot \tilde{e}\right)^T$. Therefore, we have $\partial\alpha^*/\partial\tilde{x} = \left(\lambda_2^2/b\right) \cdot \tilde{e}^T - \left(\lambda_1/\sqrt{ab}\right) \cdot \tilde{q}^T \in \mathbb{R}^{1 \times d}$.

By substitution into H and multiplying out we see that

$$\begin{aligned} H &= \sqrt{\tfrac{b}{a}}\left(E_d - \tfrac{b}{\lambda_2^2 a}\tilde{q}\tilde{q}^T\right)\left(E_d - \tfrac{\lambda_2^2}{b}\tilde{e}\tilde{e}^T + \tfrac{\lambda_1}{\sqrt{ab}}\tilde{e}\tilde{q}^T\right) \\ &= \sqrt{\tfrac{b}{a}}\Big(E_d - \tfrac{\lambda_2^2}{b}\tilde{e}\tilde{e}^T + \tfrac{\lambda_1}{\sqrt{ab}}\tilde{e}\tilde{q}^T \\ &\qquad - \tfrac{b}{\lambda_2^2 a}\tilde{q}\tilde{q}^T + \tfrac{\lambda_1}{\sqrt{ab}}\tilde{q}\tilde{e}^T - \tfrac{\lambda_1^2}{\lambda_2^2 a}\tilde{q}\tilde{q}^T\Big), \end{aligned}$$

where $\tilde{q}\tilde{q}^T\tilde{e}\tilde{e}^T = \tilde{q}\left(\tilde{q}^T\tilde{e}\right)\tilde{e}^T = \lambda_1\sqrt{a/b} \cdot \tilde{q}\tilde{e}^T$ and $\tilde{q}\tilde{q}^T\tilde{e}\tilde{q}^T = \lambda_1\sqrt{a/b} \cdot \tilde{q}\tilde{q}^T$ have been used. The claim then follows with $b/\lambda_2^2 a + \lambda_1^2/\lambda_2^2 a = d/a$ and $\tilde{q} = \sqrt{a/b} \cdot \tilde{p}$. □

The gradient has a particular simple form, as it is essentially a scaled identity matrix with additive combination of scaled dyadic products of simple vectors. In the situation where not the entire gradient but merely its product with an arbitrary vector is required (as for example in conjunction with the backpropagation algorithm), simple vector operations are already enough to compute the product:

Corollary 7 *Algorithm 4 computes the product of the gradient of the sparseness projection with an arbitrary vector in time and space linear in the problem dimensionality n.*

Proof Let $y \in \mathbb{R}^n$ be an arbitrary vector in the situation of Theorem 6, and define $\tilde{y} := V_I y \in \mathbb{R}^d$. Then one obtains

Algorithm 4: Product of the gradient of the projection onto T with an arbitrary vector

```
Input: A point $y \in \mathbb{R}^n$, target norms $\lambda_1, \lambda_2 \in \mathbb{R}$, and the results
       of Algorithm 3: Projection $p = \operatorname{proj}_T(x)$, numbers
       $\ell_1, \ell_2^2 \in \mathbb{R}$ and $d \in \mathbb{N}$.
Output: $z := \left(\partial/\partial x\ \operatorname{proj}_T(x)\right) \cdot y \in \mathbb{R}^n$.
   // Scan and slice input vectors.
1  $j := 0$; $\tilde{p} \in \{0\}^d$; $\tilde{y} \in \{0\}^d$; $\mathrm{sum}_{\tilde{y}} := 0$; $\mathrm{scp}_{\tilde{p},\tilde{y}} := 0$;
2  for $i := 1$ to $n$ where $p_i > 0$ do
3      $j := j + 1$; $\tilde{p}_j := p_i$; $\tilde{y}_j := y_i$;
4      $\mathrm{sum}_{\tilde{y}} := \mathrm{sum}_{\tilde{y}} + \tilde{y}_j$; $\mathrm{scp}_{\tilde{p},\tilde{y}} := \mathrm{scp}_{\tilde{p},\tilde{y}} + \tilde{p}_j \cdot \tilde{y}_j$;
5  end
   // Compute gradient product in sliced
      space.
6  $a := d\ell_2^2 - \ell_1^2$; $b := d\lambda_2^2 - \lambda_1^2$;
7  $\tilde{y} := \sqrt{b/a} \cdot \tilde{y}$;
8  $\tilde{y} := \tilde{y} + \sqrt{1/ab} \cdot \left(\lambda_1 \cdot \mathrm{sum}_{\tilde{y}} - d \cdot \mathrm{scp}_{\tilde{p},\tilde{y}}\right) \cdot \tilde{p}$;
9  $\tilde{y} := \tilde{y} + \sqrt{1/ab} \cdot \left(\lambda_1 \cdot \mathrm{scp}_{\tilde{p},\tilde{y}} - \lambda_2^2 \cdot \mathrm{sum}_{\tilde{y}}\right) \cdot \tilde{e}$;
   // Un-slice to yield final result.
10 $j := 0$; $z \in \{0\}^n$;
11 for $i := 1$ to $n$ where $p_i > 0$ do $j := j + 1$; $z_i := \tilde{y}_j$;
12 return $z$;
```

$$\begin{aligned} G\tilde{y} = \sqrt{\tfrac{b}{a}}\tilde{y} &+ \tfrac{1}{\sqrt{ab}}\left(\lambda_1 \cdot \tilde{e}^T\tilde{y} - d \cdot \tilde{p}^T\tilde{y}\right) \cdot \tilde{p} \\ &+ \tfrac{1}{\sqrt{ab}}\left(\lambda_1 \cdot \tilde{p}^T\tilde{y} - \lambda_2^2 \cdot \tilde{e}^T\tilde{y}\right) \cdot \tilde{e} \in \mathbb{R}^d. \end{aligned}$$

Algorithm 4 starts by computing the sliced vectors $\tilde{p}$ and $\tilde{y}$, and computes "$\mathrm{sum}_{\tilde{y}}$" which equals $\tilde{e}^T\tilde{y}$ and "$\mathrm{scp}_{\tilde{p},\tilde{y}}$" which equals $\tilde{p}^T\tilde{y}$ after Line 5. It then computes a and b using the numbers output by Algorithm 3. From Line 7 to Line 9, the product $G\tilde{y}$ is computed in-place by scaling of $\tilde{y}$, adding a scaled version of $\tilde{p}$, and adding a scalar to each coordinate. Since $\left(\partial/\partial x\ \operatorname{proj}_T(x)\right) \cdot y = V_I^T G\tilde{y}$, it just remains to invert the slicing. The complexity of the algorithm is clearly linear, both in time and space. □

It has not escaped our notice that Corollary 7 can also be used to determine the eigensystem of the projection's gradient, which may prove useful for further analysis of gradient-based learning methods involving the sparseness-enforcing projection operator.

References

Aharon, M., Elad, M., & Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, *54*(11), 4311–4322.

Bauer, F., & Memisevic, R. (2013). Feature grouping from spatially constrained multiplicative interaction. In *Proceedings of the International Conference on Learning Representations*. arXiv:1301.3391v3.

Bell, A. J., & Sejnowski, T. J. (1997). The "independent components" of natural scenes are edge filters. *Vision Research*, *37*(23), 3327–3338.

Bertsekas, D. P. (1999). *Nonlinear programming* (2nd ed.). Belmont: Athena Scientific.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Blackford, L. S., et al. (2002). An updated set of basic linear algebra subprograms (BLAS). *ACM Transactions on Mathematical Software*, *28*(2), 135–151.

Bottou, L., & LeCun, Y. (2004). Large scale online learning. In *Advances in Neural Information Processing Systems* (Vol. 16, pp. 217–224).

Bredies, K., & Lorenz, D. A. (2008). Linear convergence of iterative soft-thresholding. *Journal of Fourier Analysis and Applications*, *14*(5–6), 813–837.

Coates, A., & Ng, A. Y. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the International Conference on Machine Learning* (pp. 921–928).

Deutsch, F. (2001). *Best approximation in inner product spaces*. New York: Springer.

Dong, W., Zhang, L., Shi, G., & Wu, X. (2011). Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *IEEE Transactions on Image Processing*, *20*(7), 1838–1857.

Donoho, D. L. (1995). De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, *41*(3), 613–627.

Donoho, D. L. (2006). For most large underdetermined systems of linear equations the minimal ℓ_1-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, *59*(6), 797–829.

Duarte-Carvajalino, J. M., & Sapiro, G. (2009). Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. *IEEE Transactions on Image Processing*, *18*(7), 1395–1408.

Eckart, C., & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, *1*(3), 211–218.

Elad, M. (2006). Why simple shrinkage is still relevant for redundant representations? *IEEE Transactions on Information Theory*, *52*(12), 5559–5569.

Foucart, S., & Rauhut, H. (2013). *Mathematical introduction to compressive sensing*. New York: Birkhäuser.

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., et al. (2009). *GNU scientific library reference manual* (3rd ed.). Bristol: Network Theory Ltd.

Gharavi-Alkhansari, M., & Huang, T. S. (1998). A fast orthogonal matching pursuit algorithm. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (Vol. III, pp. 1389–1392).

Goldberg, D. (1991). What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, *23*(1), 5–48.

Hawe, S., Seibert, M., & Kleinsteuber, M. (2013). Separable dictionary learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 438–445).

Hoggar, S. G. (2006). *Mathematics of digital images: Creation, compression, restoration, recognition*. Cambridge: Cambridge University Press.

Horev, I., Bryt, O., & Rubinstein, R. (2012). Adaptive image compression using sparse dictionaries. In *Proceedings of the International Conference on Systems, Signals and Image Processing* (pp. 592–595).

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, *5*, 1457–1469.

Hoyer, P. O., & Hyvärinen, A. (2000). Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems, 11*(3), 191–210.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, *148*(3), 574–591.

Hurley, N., & Rickard, S. (2009). Comparing measures of sparsity. *IEEE Transactions on Information Theory*, *55*(10), 4723–4741.

Hyvärinen, A. (1999). Sparse code shrinkage: Denoising of nongaussian data by maximum likelihood estimation. *Neural Computation*, *11*(7), 1739–1768.

Hyvärinen, A., & Hoyer, P. O. (2000). Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Computation*, *12*(7), 1705–1720.

Hyvärinen, A., Hoyer, P. O., & Inki, M. (2001). Topographic independent component analysis. *Neural Computation*, *13*(7), 1527–1558.

Hyvärinen, A., Hurri, J., & Hoyer, P. O. (2009). *Natural image statistics–A probabilistic approach to early computational vision*. London: Springer.

Jones, J. P., & Palmer, L. A. (1987). An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, *58*(6), 1233–1258.

Kavukcuoglu, K., Ranzato, M., Fergus, R., & LeCun, Y. (2009). Learning invariant features through topographic filter maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1605–1612).

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480.

Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T.-W., & Sejnowski, T. J. (2003). Dictionary learning algorithms for sparse representation. *Neural Computation*, *15*(2), 349–396.

Laughlin, S. B., & Sejnowski, T. J. (2003). Communication in neuronal networks. *Science*, *301*(5641), 1870–1874.

Liu, J., & Ye, J. (2009). Efficient Euclidean projections in linear time. In *Proceedings of the International Conference on Machine Learning* (pp. 657–664).

Lopes, M. E. (2013). Estimating unknown sparsity in compressed sensing. In *Proceedings of the International Conference on Machine Learning* (pp. 217–225).

Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2009a). Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning* (pp. 689–696).

Mairal, J., Bach, F., Ponce, J., Sapiro, G., & Zisserman, A. (2009b). Non-local sparse models for image restoration. In *Proceedings of the International Conference on Computer Vision* (pp. 2272–2279).

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*(4), 308–313.

Neudecker, H. (1969). Some theorems on matrix differentiation with special reference to Kronecker matrix products. *Journal of the American Statistical Association*, *64*(327), 953–963.

Olmos, A., & Kingdom, F. A. A. (2004). A biologically inspired algorithm for the recovery of shading and reflectance images. *Perception*, *33*(12), 1463–1473.

Olshausen, B. A. (2003). Learning sparse, overcomplete representations of time-varying natural images. In *Proceedings of the International Conference on Image Processing* (Vol. I, pp. 41–44).

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*(6583), 607–609.

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, *37*(23), 3311–3325.

Potluru, V. K., Plis, S. M., Le Roux, J., Pearlmutter, B. A., Calhoun, V. D., & Hayes, T. P. (2013). Block coordinate descent for sparse NMF. In *Proceedings of the International Conference on Learning Representations*. arXiv:1301.3527v2.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes: The art of scientific computing* (3rd ed.). Cambridge: Cambridge University Press.

Rigamonti, R., Sironi, A., Lepetit, V., & Fua, P. (2013). Learning separable filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2754–2761).

Ringach, D. L. (2002). Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of Neurophysiology*, *88*(1), 455–463.

Rodgers, J. L., & Nicewander, W. A. (1988). Thirteen ways to look at the correlation coefficient. *The American Statistician*, *42*(1), 59–66.

Rozell, C. J., Johnson, D. H., Baraniuk, R. G., & Olshausen, B. A. (2008). Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, *20*(10), 2526–2563.

Skretting, K., & Engan, K. (2010). Recursive least squares dictionary learning algorithm. *IEEE Transactions on Signal Processing*, *58*(4), 2121–2130.

Skretting, K., & Engan, K. (2011). Image compression using learned dictionaries by RLS-DLA and compared with K-SVD. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 1517–1520).

Society of Motion Picture and Television Engineers (SMPTE). (1993). Recommended practice RP 177–193: Derivation of basic television color equations.

Theis, F. J., Stadlthanner, K., & Tanaka, T. (2005). First results on uniqueness of sparse non-negative matrix factorization. In *Proceedings of the European Signal Processing Conference* (Vol. 3, pp. 1672–1675)

Thom, M., & Palm, G. (2013). Sparse activity and sparse connectivity in supervised learning. *Journal of Machine Learning Research*, *14*, 1091–1143.

Tošić, I., Olshausen, B. A., & Culpepper, B. J. (2011). Learning sparse representations of depth. *IEEE Journal of Selected Topics in Signal Processing*, *5*(5), 941–952.

Traub, J. F. (1964). *Iterative methods for the solution of equations*. Englewood Cliffs: Prentice-Hall.

van Hateren, J. H., & Ruderman, D. L. (1998). Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. *Proceedings of the Royal Society B*, *265*(1412), 2315–2320.

Wang, Z., & Bovik, A. C. (2009). Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE Signal Processing Magazine*, *26*(1), 98–117.

Watson, A. B. (1994). Image compression using the discrete cosine transform. *The Mathematica Journal*, *4*(1), 81–88.

Willmore, B., & Tolhurst, D. J. (2001). Characterizing the sparseness of neural codes. *Network: Computation in Neural Systems*, *12*(3), 255–270.

Wilson, D. R., & Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, *16*(10), 1429–1451.

Yang, J., Wang, Z., Lin, Z., Cohen, S., & Huang, T. (2012). Coupled dictionary training for image super-resolution. *IEEE Transactions on Image Processing*, *21*(8), 3467–3478.

Yang, J., Wright, J., Huang, T., & Ma, Y. (2010). Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, *19*(11), 2861–2873.

Zelnik-Manor, L., Rosenblum, K., & Eldar, Y. C. (2012). Dictionary optimization for block-sparse representations. *IEEE Transactions on Signal Processing*, *60*(5), 2386–2395.