Leopold-Franzens-Universität Innsbruck

Department of Mathematics

Bachelor Thesis

Comparison of ℓ_4-norm minimization to alternating projection methods for dictionary learning

Christina Schwaiger
Matriculation number: 11808809
christina.schwaiger@student.uibk.ac.at

supervised by
Univ. Prof. Dr. Karin Schnass

Innsbruck, October 8, 2021

Abstract

The aim of dictionary learning is to learn a set of vectors based on some training data so that every signal can be approximated by a linear combination of these atoms. The goal of this thesis is to study the dictionary learning method presented by [QZL$^+$19], where ℓ_4-norm minimization is used to solve the dictionary leaning problem and recover the atoms of the dictionary one after another. In [QZL$^+$19] it was shown that each solution of the proposed optimization problem is close to one column of the dictionary if certain constraints hold. It was also shown that these solutions are global minimizers and that the algorithm converges globally. We want to examine if we can recover a dictionary and some sparse coefficients, which approximate the signals given, by using this method. We therefore implemented this method and tested it with synthetic and image data. During the implementation phase of the algorithm some questions arose, concerning the proper initialization of the starting values for the method. We therefore came up with different approaches on how to get good initialization values. Concerning the synthetically generated data, the method proposed in [QZL$^+$19] worked well, if all the theoretical constraints are satisfied, despite having a slightly high computational effort. When taking a look at the experiments, we could see that these constraints are necessary for the algorithm to recover an appropriate dictionary. If they are not fulfilled, we were not able to recover some atoms of the dictionary. We then compared the algorithm with two alternating projection methods by considering the dictionaries learned on image data. We used here the algorithms K-SVD and ITKrM, which are both widely used for learning dictionaries on image patches. In this setting when the constraints presented in [QZL$^+$19] did not hold, the presented method in [QZL$^+$19] did not work well. Therefore it might be more preferable to use alternating projection methods in the field of dictionary learning with images.

Keywords: dictionary learning, non-convex optimization, manifold optimization, overcomplete representation, implementation.

Contents

Declaration

By my own signature I declare that I produced this work as the sole author, working independently, and that I did not use any sources and aids other than those referenced in the text. All passages borrowed from external sources, verbatim or by content, are explicitly identified as such.

Signature: _____ Date: _____

Chapter 1

Introduction

When sampling smooth signals we represent them as a sum of some Dirac delta functions or by using a Dirac-comb and of course we can represent each signal sampled d times as vector in $\mathbb{R}^d$. This vector then has a representation as linear combination of the standard basis vectors $e_i \in \mathbb{R}^d$ for some constants c_i. Also if we make a basis change we are able to represent the signal by taking some other constants $\tilde{c}_i$. Most likely we will need to sum lots of scaled vectors $\tilde{c}_i\tilde{e}_i$ in order to represent our signal properly. In most signal processing tasks however we require to have another representation of the signals, which captures the characteristics of the signal better and is therefore more useful. For example for denoising signals the representation should separate the signal and the noise. In the field of compressed sensing or signal recovery we want to find a representation or approximation of the signal which uses only a few coefficients, also called a sparse representation. Especially if the signals are of high dimension, we want to express these signals by using a small number of linear combinations of a specific signal representation system. If we normalize the columns of the sparse representation matrix we call it dictionary. So we define a dictionary $A = (a_1, \dots, a_m) \in \mathbb{R}^{d\times m}$ to be a matrix which contains normalized basic signals a_i, also called atoms, which are saved column-wise. There are no further restrictions on the dictionary, e.g. it is not required to have orthogonal columns. Therefore some columns might be linear combinations of some other columns. With this dictionary A we want to approximate the given signals or data. More specifically we want to write the signals as a linear combination of a few columns of A. This is also known as sparse approximation. We have a fixed, known and predefined dictionary A and want to decompose the signal. For a single signal $y_i \in \mathbb{R}^d$ we can express the sparse approximation problem as optimization problem, where we want to find a vector x_i. This vector x_i should have as many zero entries as possible and the product of the matrix and the vector x_i should approximate the signal y_i. This means we have the following problem

$$\min_{x_i} \|x_i\|_0 \quad \text{such that} \quad y_i = Ax_i, \tag{1.1}$$

where $\|\cdot\|_0$ denotes the ℓ_0-norm, which counts the nonzero entries. A widely used relaxation to this sparse approximation problem is to consider the ℓ_1 norm instead of the pseudo-norm ℓ_0. The vector x_i is called s-sparse in this sense, for some $s \in \mathbb{N}$, if it has only up to s entries that are not zero. The goal of this problem is to achieve the relation $s \ll d$, such that this vector x_i has much more zero than nonzero entries. This is often not possible. Therefore we substitute the equality constraint in Equation (1.1) by an approximation expressed as

$$y_i = Ax_i + \eta \tag{1.2}$$

for η small. Equation (1.1) can also be interpreted in the sense, that we want to express our signal as compactly as possible. Further, the fact that the dictionary can also have linearly dependent columns, allows us to reduce the sparsity level s. Here we keep the rule *'The sparser the representation, the better the dictionary'* in mind [Sch15a]. For several signals, collected inside a matrix $Y = (y_1, \dots, y_p) \in \mathbb{R}^{d \times p}$, we formulate the sparse approximation problem as

$$\min_X \|Y - AX\|_F^2 = \min_X \sum_{i=1}^{p} \|y_i - Ax_i\|_2^2 \quad \text{such that} \quad \forall i: \ \|x_i\|_2 \leq s, \tag{1.3}$$

with $\|\cdot\|_F$ denoting the Frobenius norm and $X = (x_1, \dots, x_p) \in \mathbb{R}^{m \times p}$ being column-wisely s-sparse for some fixed $s \in \mathbb{N}$, where s is as small as possible. There are several good ways how to define the dictionary A that is used and held fixed in the sparse approximation problem. For example one could take a discrete cosine transformation (DCT), a Fourier or a Wavelet transformation as dictionary. Further there are many sparse approximation algorithms that find the sparse coefficients X by solving problem (1.3) for a given and fixed dictionary A. We are not going to cover them more specifically. For a good overview, we refer to [TW10].

By now, as we have clarified what a dictionary and sparse approximation are, we can also define the optimization problem of dictionary learning. In the following we allow the dictionary A to vary as well and learn it from the data. So we take Equation (1.3) and search for the matrices A and X which fulfill the approximation of the data as well as the sparsity constraint for a given sparsity level s. We have the following problem

$$\min_{A \in \mathcal{A}, X \in \mathcal{X}} \sum_{i=1}^{p} \|y_i - Ax_i\|_2^2 = \min_{A \in \mathcal{A}, X \in \mathcal{X}} \|Y - AX\|_F^2 \tag{1.4}$$

with $\mathcal{A} = \{A \in \mathbb{R}^{d \times m} : \|a_i\|_2 = 1 \text{ for all } 1 \leq i \leq m\}$ and $\mathcal{X} = \{X \in \mathbb{R}^{m \times p} : \|x_i\|_0 < s \text{ for all } 1 \leq i \leq p\}$. In the optimization problem in (1.4), in contrast to the sparse approximation problem in (1.3), the dictionary is also dependent on the data. This property enables the sparsity level s to be smaller compared to the sparsity level needed in the sparse approximation problem. So more entries of the columns of the sparse coefficients X are zero. Therefore using learned dictionaries for the representation of the data leads to better results in many applications [SBZJ14, EA06, BE08, MBP12].
In this thesis, we will mainly consider dictionary learning in the overcomplete case (ODL), meaning that the dictionary or matrix A is of rectangle form and we have more columns than rows, so $m > d$. The advantage of these dictionary learning methods is that they provide greater flexibility in saving column-vectors in the dictionary. So we can also have columns in the dictionary that can be represented by a linear combination of some other columns. This property leads to a better chance of approximating the signals with a smaller sparsity level s. The number of nonzero entries in each column of the sparse coefficients X is reduced.
Summarizing, we use dictionary learning in order to find some matrices A and X such that the error $\|Y - AX\|_F^2$ is small and X has many nonzero entries. In order to solve this optimization problem we can use alternating projection methods or try to find the dictionary A in another way. When using an alternating projection method we alternate between recovering the sparse coefficients of X and modifying the dictionary A according to X. In this thesis we consider the paper [QZL$^+$19] which deals with overcomplete dictionary learning. The paper by [QZL$^+$19] is based on the approach of [SQW17] for complete dictionaries, where $m = d$. The difference of the method in [QZL$^+$19] to other dictionary learning methods is that the

individual columns of the matrix A are found directly by independent trials without the need of the sparse coefficients X. When all columns of the dictionary A could be recovered, any sparse approximation algorithm can be used to find the sparse coefficients X. The algorithm that was considered in the implementation and for the experiments is a greedy method named Orthogonal Matching Pursuit (OMP).

Contribution The goal of this thesis is to implement and evaluate the method presented by [QZL$^+$19]. We started by analyzing the paper [QZL$^+$19] and checking the statements and the proofs for correctness again. Then the algorithm presented in [QZL$^+$19] was implemented and several experiments were made concerning the ability to recover the dictionary and the sparse coefficients. We also considered the stability of the algorithm if not all the assumptions made in [QZL$^+$19] are perfectly met and compared the learned dictionary to the ones we could recover by using the alternating projection methods K-SVD and ITKrM. This was done by using synthetically generated data as well as image data.

Outline The next section deals with our notational conventions. To get a better understanding of dictionary learning algorithms, the three alternating projection methods ITKrM, MOD and K-SVD and their ideas will be discussed in more detail in Chapter 3. The theory of [QZL$^+$19] is presented in Chapter 4. The algorithm description referring to this new method can be found in Chapter 5. The results of the practical comparison can be found in Chapter 6. The Python code from [PSKK20] was used for the implementation and the methods to be investigated were added. The image used for the comparison was the famous computer vision image called *Barbara*.

Chapter 2

Notation

Lower-case letters denote vectors $v \in \mathbb{R}^d$ or numbers $c \in \mathbb{R}$ depending on the given context whereas upper-case letters stand for matrices $X \in \mathbb{R}^{d \times d}$. We write $\|x\|_p$ to determine the standard ℓ_p-norm of a vector, and $\|x\|_0$ to denote the number of nonzero entries. $\|X\|_F$ denotes the Frobenius norm, which is defined as $\sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2}$ for $X \in \mathbb{R}^{m \times n}$. The Hadamard product is denoted here by $\circ$ and represents the point-wise product between two vectors $(x \circ y)_i = (x_i y_i)$ or two matrices $(X \circ Y)_{ij} = (x_{ij} y_{ij})$ of the same dimensions. To denote entries of a specific vector x or a matrix X we write x_1 for the first entry and $X_{1,2}$ for the second entry in the first row of the matrix. A^* stands for the conjugate transpose of a matrix A. $\mathbb{I}$ stands for the identity matrix and $\mathbb{E}$ stands for the expectation. $A_{\mathcal{J}}$ symbolizes the restriction of dictionary A to atoms that are inside a specific set $\mathcal{J}$ and $P(A_{\mathcal{J}})$ indicates the orthogonal projection onto the span of the dictionary vectors in $\mathcal{J}$. $|\mathcal{J}|$ denotes the cardinality of a given set $\mathcal{J}$. The set $\mathcal{I}$ in the following indicates the set of indices k of the nonzero entries of x_i and is therefore also referred to as the support. The cardinality of this set $|\mathcal{I}| = s$ is equal to the sparsity level. The function $\operatorname{sign}(\cdot)$ returns the sign of the expression in brackets. The indicator function χ of a set $\mathcal{M}$, written as $\chi(\mathcal{M}, x)$, is one if $x \in \mathcal{M}$ and zero otherwise. The Landau symbol $\mathcal{O}$ characterizes the asymptotic behavior or growth of a function, where $f(t) = \mathcal{O}(g(t))$ means

$$\lim_{t \to 0/\infty} \frac{f(t)}{g(t)} = C < \infty.$$

We will use it as a measurement for the computational complexity of an algorithm or evaluation.

Chapter 3

Alternating projection algorithms

Dictionary learning methods mostly differ according to the way the dictionary is found [SBZJ14]. Most likely, the dictionary learning problem is solved by using an alternating projection method to recover or learn the dictionary A and the sparse coefficient matrix X. These alternating projection methods alternate between minimizing the expression in Equation (1.3) with respect to the sparse coefficients X while fixing the dictionary A in the meantime and updating the dictionary based on these sparse representations X [AEB06, NS17a]. An initial dictionary A has to be given, such that we can start right away with the first alternating step. By following this scheme, we want to find solutions of the following optimization problem

$$\begin{aligned} &\min_{A,X} \|Y - AX\|_F^2 = \min_{A,X} \sum_{i=1}^{n} \|y_i - Ax_i\|_2^2 \\ &\text{such that} \quad \forall i : \|x_i\|_0 \leq S \quad \text{and} \quad A \in \mathbb{R}^{d \times m}. \end{aligned} \tag{3.1}$$

Alternating projection methods are widely used in practice, because they are numerically stable, easy and achieve very good results. The alternating projection methods we are going to consider here are MOD, K-SVD and ITKrM. As stated in [LBM11], the performance of K-SVD and MOD is similar and therefore we do not consider the MOD algorithm for the practical results later on in Chapter 6.

3.1 MOD

The algorithm MOD uses Orthogonal Matching Pursuit (OMP) to solve the sparse approximation problem (1.3) in the first alternating step. The OMP algorithm iteratively selects the column of the dictionary A which has the highest inner product with the current residual. Then the signal vector estimation x_i is updated and the new residual is computed based on the updated signal x_i and the dictionary A [MZ93]. The dictionary is then subject of a simple update process. We denote the error that we make by decomposing signal y_i as

$$e_i = y_i - Ax_i. \tag{3.2}$$

If the sparse coefficient matrix X stays fixed, the dictionary should be updated such that the error

$$\|E\|_F^2 = \|(e_1, \ldots, e_m)\|_F^2 = \|Y - AX\|_F^2 \tag{3.3}$$

is minimized. Deriving Equation (3.3) with respect to A gives

$$(Y - AX)X^T = 0$$

leading to

$$\tilde{A} = YX^T(XX^T)^{-1},$$

where $\tilde{A}$ denotes the new iterate. Afterwards the columns of the iterate $\tilde{A}$ have to be normalized again [AEB06, EAH99]. By construction, the error is minimized by considering the new iterate $\tilde{A}$ instead of A. This iterate minimizes the error E. Afterwards we set $A = \tilde{A}$ and start at the first alternating step again. This iteration ends after the error stays relatively constant or a fixed number of iterations has been performed [EAH99]. Typically only a few iterations are needed in order to converge to a dictionary with whom we can approximate the data accurately with a small error [RBE10]. The difference to the K-SVD algorithm described below is that the whole set of atoms in the dictionary is updated at the same time whereas in the K-SVD algorithm each dictionary atom is updated separately.

3.2 K-SVD

Similar to MOD, in the first alternating step of K-SVD we want to find some sparse coefficients X. So we try to minimize Equation (3.1) with respect to X by using OMP while fixing the dictionary A in the meantime. For the dictionary update we update each column after another and fix the other columns as well as the coefficient matrix X during this update process. In the i-th update for the atom a_i of the dictionary A, the entries of the columns of X are evaluated that use this column vector of A, so we define

$$\mathcal{W}_i = \{k : x_i(k) \neq 0\},$$

where $x_i(k)$ denotes the k-th entry of the row x_i. Therefore we only use those signals for the update of the dictionary that make use of the corresponding column vector in the dictionary. We fix everything but a_i and the nonzero locations of x_i. Then, as above, the approximation error $E_i = Y - \sum_{j \neq i} a_j x_j$, where the column a_i is not considered, is determined. In order to not lose the sparsity constraint, we will consider the error $E_i^R = (E_i)_{\mathcal{W}_i}$ in the following. This error E_i^R only takes into account those columns whose indices occur in $\mathcal{W}_i$. We want to minimize the error, which can now be written as

$$\|E_i^R - a_i x_i^R\|_F^2.$$

Because of $a_i x_i^R$ being a rank-1 matrix this problem is equivalent to finding the closest rank-1 approximation to E_i^R [SBZJ14]. To solve the problem we use the Singular-Value-Decomposition (SVD). We decompose E_i^R into

$$E_i^R = U\Sigma V^T$$

with SVD. The new column $\tilde{a}_i$ of the dictionary should now be selected as the first column of U, the others remain fixed. The first column of V multiplied by the entry $\Sigma_{1,1}$ describes the new coefficients x_i^R [AEB06]. Therefore the associated row vectors of the sparse coefficient matrix are also updated. The amount of nonzero entries of each column of X is hereby not increasing and we do not violate the constraint of the sparsity level [SBZJ14].

3.3 ITKrM

ITKrM uses thresholding for the sparse approximation of the signals. For the update of the dictionary one uses residual averages. ITKrM can also be viewed as a fixed point iteration,

where the dictionary A describes the fixed point [NS17a]. The thresholding is done by computing the set $\mathcal{I}_i^t$, also called the sparse support, for every signal y_i in Y. This set $\mathcal{I}_i^t$ consists of the indices of the s largest inner products in magnitude of the atoms of A and the signals themselves and is defined in the following way

$$\mathcal{I}_i^t = \arg\max_{|\mathcal{I}|=s} \|A_{\mathcal{I}}^* y_i\|_1 = \arg\max_{|\mathcal{I}|=s} \sum_{k\in\mathcal{I}} |\langle a_k, y_i\rangle|.$$

Afterwards the dictionary update is done by computing the residual means. For all k and $Y \in \mathbb{R}^{d\times p}$ we calculate

$$\tilde{a}_k = \frac{1}{p} \sum_{\substack{i \text{ s.t.} \\ k\in\mathcal{I}_i^t}} \operatorname{sign}(\langle a_k, y_i\rangle) \cdot (y_i - P(A_{\mathcal{I}_i^t}) y_i + P(a_k) y_i).$$

Afterwards this $\tilde{a}_k$ is normalized and makes up the updated column of the dictionary A [Sch15a]. The algorithm has some advantages. First of all the computational costs of thresholding are lower than those of OMP and also the residual averages are faster to compute than SVD. In addition, one signal is processed after the other, which also enables parallelization [Sch15a]. The algorithm works sequentially meaning that we only need up to two times the memory space of the dictionary whereas it is much more in the case of K-SVD.

Chapter 4

ℓ_4-norm minimization

But there are also other dictionary learning methods where e.g. only the dictionary A is recovered and we find the sparse coefficients X by using a sparse approximation method. One recently proposed dictionary learning method is the one in [QZL$^+$19] that we want to describe in the following. The difference to other papers dealing with dictionary learning methods is the approach of not recovering the sparse coefficients of the representation matrix X but rather directly finding columns of the dictionary A. So if we compare this method to the alternating projection methods in Chapter 3, we leave out the first alternating step. In this step we searched for the sparse coefficients X, with which we then updated the dictionary A in an algorithm-dependent manner. The goal of [QZL$^+$19] is to prove that we are able to recover one column of the dictionary after another by solving a given optimization problem. In the paper some theoretical assumptions are made such that given statements can be proven. Before we define the optimization problem we need to discuss the assumptions made on the dictionary A and the sparse coefficients X.

- The first assumption on the dictionary is that $A \in \mathbb{R}^{d\times m}$ is a unit norm tight frame (UNTF), which means that

$$\frac{d}{m}AA^T = \mathbb{I}, \quad \|a_i\|_2 = 1 \quad \text{for} \quad 1 \leq i \leq m \tag{4.1}$$

  holds.

- Further, the coherence of the dictionary $\mu(A) \ll 1$ defined as

$$\mu = \mu(A) = \max_{1\leq i\neq j\leq m} |\langle a_i, a_j\rangle| \in (0,1) \tag{4.2}$$

  should be very small. This variable μ describes the correlation of the columns of the matrix A. A small coherence means that the columns are almost orthogonal to each other. As we deal with ODL, the vectors cannot be orthogonal to each other as we have more than $d < m$ column vectors. But it is possible that these m column vectors are close to an orthonormal basis. Therefore, a small coherence can also be interpreted as near-orthogonality condition in the overcomplete case. One can also take the definition of the scalar product of denoting the angle between two vectors, where orthogonal vectors span a 90° angle. When having small coherence we do not achieve a 90° angle between all vectors but still at least another large angle that is nearly perpendicular.

- The assumption made on the entries of the coefficient matrix $X \sim_{i.i.d.} BG(s)$ is that they are Bernoulli-Gaussian distributed with parameter s. This means the sparse coefficients

are the Hadamard product of a Bernoulli distributed variable and a Gaussian distributed variable. They are defined as

$$X = B \circ G, \quad B_{ij} \sim_{i.i.d.} Ber(s), \quad G_{ij} \sim_{i.i.d.} \mathcal{N}(0,1), \tag{4.3}$$

with the parameter s, as aforementioned, denoting the sparsity level of X.

4.1 Problem formulation and motivation

The dictionary learning problem is then formulated as ℓ_4-norm maximization of the function ϕ_{DL} defined as

$$\max_q \phi_{DL}(q) = \frac{c_{DL}}{p}\|q^T Y\|_4^4 = \frac{c_{DL}}{p}\|q^T AX\|_4^4 \quad \text{such that} \quad \|q\|_2 = 1. \tag{4.4}$$

$c_{DL} = \frac{-1}{3s(1-s)}$ is a normalizing constant that ensures that an expectation used in the proofs of [QZL+19] is 1. In the following we will sometimes refer to this problem as ℓ_4-norm minimization as we can consider the negative value of the function and search for the minimum value. Especially when we come to algorithmic solutions there are more optimization algorithms that find the minimum value of a function than the maximum.
At first sight one may wonder why the argument that maximizes the given function ϕ_{DL} should solve our problem and be near a column of the dictionary A. Therefore we want to give some motivation for this specific function. We assume that the assumptions presented above hold, so the dictionary has low coherence and is a unit norm tight frame. Assuming that

$$\mathbb{E}_X[\phi_{DL}(q)] = \frac{1}{4}\|A^T q\|_4^4 - \frac{s}{2(1-s)}\frac{m^2}{d^2}, \tag{4.5}$$

holds, as proven in [QZL+19], minimization Problem (4.4) can be described as the maximization of the ℓ_4-norm of $A^T q$ over the d-dimensional sphere for p large. In the following we will write $\zeta(q) = A^T q$ for abbreviation and better understanding. We now assume that we have found a solution q^*, which approximates one column a_i of the dictionary A. If without loss of generality $q^* \approx a_1$, then

$$\zeta(q^*) = A^T q^* \approx (\|a_1\|^2, a_1^T a_2, \ldots, a_1^T a_m)^T = (1, \delta_2, \ldots \delta_m)^T, \tag{4.6}$$

where $\delta_i < \mu \ll 1$ due to Assumption (4.2). We define

$$\rho(\zeta(q)) = \frac{\zeta_{(1)}}{\zeta_{(2)}}, \tag{4.7}$$

as a measure of spikiness, where $|\zeta_{(1)}| \geq |\zeta_{(2)}| \geq \ldots \geq |\zeta_{(m)}|$ symbolize the ordered entries of $\zeta(q)$. So if we maximize the ℓ_4-norm over the sphere, we also increases the spikiness of $\zeta(q)$. We also know that the spikier a vector q is, the larger $\|\zeta(q)\|_4^4$ gets. Therefore the spikiest vectors would be solutions of the maximization problem in (4.4). This can easily be seen by considering the unit circle in $\mathbb{R}^2$, so where $\|x\|_2 = 1$. The points on the axes are the spikiest ones and also the norm $\|x\|_4^4$ is maximal for this points $(0,1), (1,0), (0,-1), (-1,0)$. This of course also applies for higher dimensions.
When taking a look at Equation (4.6) again, we expect that the argument maximizing the value of function ϕ_{DL} is close to one column of the dictionary A[QZL+19]. So we are able to recover one atom of the dictionary by this approach when the assumptions are fulfilled.

4.2 Proof sketch

Theorem The main result of [QZL+19] is that there are no spurious local minimizers over the sphere $\mathbb{S}^{d-1}$, if A and X fulfill the given assumptions in (4.1) and (4.3) as well as the condition $Y = AX$. All critical points q are with high probability either strict saddle points, which can be filtered out with the help of a negative curvature, or are close to an atom of A, where close means

$$\left\langle \frac{a_i}{\|a_i\|}, q \right\rangle \geq 1 - 5\xi^{-\frac{3}{2}}. \tag{4.8}$$

Proof idea The idea of [QZL+19] to prove the statement is to use the fact that under Assumption (4.1) and Assumption (4.3)

$$\mathbb{E}_X[\phi_{DL}(q)] = \phi_T(q) - \frac{S}{2(1-S)}\frac{m^2}{d^2} \tag{4.9}$$

with

$$\phi_T(q) = \frac{1}{4}\|A^T q\|_4^4 \tag{4.10}$$

holds. This equation tells us that if p is large enough ϕ_{DL} reduces to ϕ_T. We use this and first analyze the critical points of ϕ_T. Therefore we divide the unit sphere $\mathbb{S}^{d-1}$ into two disjoint regions and analyze the critical points in the given regions. The two disjoint regions are denoted in the following by

$$R_N(q;\xi) = \{q \in \mathbb{S}^{d-1} : \phi_T(q) \geq \xi\mu^{\frac{2}{3}}\|A^T q\|_3^2\}$$
$$R_C(q;\xi) = \{q \in \mathbb{S}^{d-1} : \phi_T(q) \leq \xi\mu^{\frac{2}{3}}\|A^T q\|_3^2\}$$

with $\xi > 0$ being a scalar. The first thing to do is to classify the critical points.

We will write $\zeta(q) = A^T q = (\zeta_1, \dots \zeta_m)$. Without loss of generality we assume that $|\zeta_1| \geq |\zeta_2| \geq \dots \geq |\zeta_m|$ for a given and fixed $q \in \mathbb{S}^{d-1}$. It is shown that with high probability over the randomness of the sparse coefficients X all points in R_N have negative curvature meaning

$$\exists v \in \mathbb{S}^{d-1} \quad \text{such that} \quad v^T \text{Hess}\phi_{DL}(q)v \leq -3\|A^T q\|_4^4\|A^T q\|_\infty^2. \tag{4.11}$$

Concerning the critical points in R_C then, with high probability we can show that there are no falsely interpreted critical points in the sense that every critical point is either near a column of the dictionary A or is a strict saddle point. So one critical point in R_C either solves the optimization problem, or can be escaped because of the negative curvature. In more detail, one can classify the critical points $q \in R_C$ into three disjoint classes considering the entries of the $\zeta = \zeta(q)$ for some fixed $q \in R_C$.

Class 1 $\forall 1 \leq i \leq m : |\zeta_i| \leq \frac{2|\beta_i|}{\alpha_i}$

Class 2 $|\zeta_1| > \frac{2|\beta_1|}{\alpha_1}$ and $|\zeta_i| \leq \frac{2|\beta_i|}{\alpha_i}$ for $i = 2, \dots, m$

Class 3 $|\zeta_i| > \frac{2|\beta_i|}{\alpha_i}$ for $i = 1, 2$

with

$$\alpha_i = \|\zeta\|_4^4 > 0 \qquad \text{and} \qquad \beta_i = \sum_{j \neq i} \langle a_i, a_j \rangle \zeta_j^3.$$

Afterwards it is shown that there are no critical points whose entries are all smaller than $\frac{2|\beta_i|}{\alpha_i}$, meaning there are no critical points in Class 1. Points satisfying the condition of Class 2 are near global minimizers and therefore near one column of the dictionary A. Regarding the last case, Class 3, it is proven that critical points fulfilling this condition are saddle points that can be escaped by negative curvature.
Afterwards we argue why it is correct to concentrate our analysis on $\phi_T = \frac{1}{4}\|A^T q\|_4^4$ instead of ϕ_{DL}. First of all for any $\delta > 0$, for $p \geq \tilde{\Omega}(\delta^{-2}\text{poly}(n))$ finitely large enough and under Assumption (4.1) and Assumption (4.3)

$$\mathbb{E}_X[\phi_{DL}(q)] = \frac{1}{4}\|A^T q\|_4^4 - \frac{s}{2(1-s)}\frac{m^2}{d^2} \tag{4.12}$$

holds. We can also say that

$$\sup_{q \in \mathbb{S}^{d-1}} \|\text{grad}\phi_{DL}(q) - \text{grad}\phi_T(q)\| \leq \delta \qquad \text{and}$$
$$\sup_{q \in \mathbb{S}^{d-1}} \|\text{Hess}\phi_{DL}(q) - \text{Hess}\phi_T(q)\| \leq \delta$$

is true. By a *perturbation analysis* we turn the analysis of the critical points of ϕ_{DL} to those of ϕ_T, as can be seen in Appendix C and D in [QZL+19]. However, these critical points have already been analyzed in the first step of this proof. The algorithm therefore guarantees the recovery of a solution which is near one column of the dictionary A.

We have seen that the vectors that maximize the value of ϕ_{DL} are near one column of the dictionary A. Therefore we can build an algorithm out of this theory by using an optimization method where we recover the argument that leads to the maximum value of the function or respectively to the minimum value of the negative function.

Chapter 5

Algorithm

We want to recall Problem (4.4), that we seek to solve in order to get a column of the dictionary A

$$\max_q \phi_{DL}(q) = \frac{c_{DL}}{p}\|q^T Y\|_4^4 = \frac{c_{DL}}{p}\|q^T AX\|_4^4 \quad \text{such} \quad \text{that} \quad \|q\|_2 = 1$$

with $c_{DL} = \frac{-1}{3s(1-s)}$ and s denoting the chosen sparsity level. As seen in Chapter 4, arguments q which maximize the value of ϕ_{DL} are close to the columns of the dictionary A that we want to recover. In order to work with the dictionary learning method, first some initialization and definitions have to be made. We need to specify the data matrix Y and define the shape of the dictionary as well as an appropriate sparsity level s.

5.1 Algorithm to recover one atom

In order to solve such an optimization problem we can use several algorithms that find the maximum value of the function and therefore recover one column of the dictionary A approximately. In [QZL+19] the so called power method was outlined to be a good choice to find the arguments of the maximum function values. The power method is designed for problems of the form

$$\min_q \phi(q) \quad \text{such that} \quad q \in \mathbb{S}^{d-1}$$

and converges very fast. In order to make use of this function we multiply ϕ_{DL} with -1. In the power method algorithm presented in Algorithm 1, we first need to define a starting value for the iterate that should then converge to the solution of the optimization problem. Afterwards we calculate the gradient of the negative function $-\phi_{DL}$ at the point of the iterate. The gradient of $-\phi_{DL}$ can be calculated in the following way

$$\begin{aligned}\nabla(-\phi_{DL})(q) &= \frac{-1}{3s(1-s)p}\sum_{k=1}^{p}(q^T Ax_k)^3(Ax_k) = \\ &= \frac{-1}{3s(1-s)p}\sum_{k=1}^{p}(q^T Y_k)^3(Y_k).\end{aligned}$$

The iterate is then updated by projecting the gradient of the iterate onto the unit sphere. We iterate until we converge to a solution, so we end the iteration e.g. after a certain number of iterations or if the iterate stays nearly the same compared to its previous value. We then have found one solution of Problem (4.4). This solution vector is close to one column of the

dictionary A according to the analysis in Section 4.

Algorithm 1 Power method

Input: data matrix $Y \in \mathbb{R}^{d\times p}$, sparsity level $s \in \mathbb{N}$, initial dictionary $A \in \mathbb{R}^{d\times m}$
  initialize the iterate $q^{(0)}$
  $k = 0$
  while not converged **do**
    compute $\nabla(-\phi_{DL})(q^{(k)})$
    $q^{(k+1)} = P_{\mathbb{S}^{d-1}}\left(-\nabla(-\phi_{DL})(q^{(k)})\right)$
    $k \leftarrow k + 1$
  end while
Output: $q^{(k)}$

Concerning the algorithmic costs, we get $\mathcal{O}(dp)$ for computing the gradient of $-\phi_{DL}$ which has to be called in every power method iteration.
The proof made in [QZL$^+$19] only showed the ability to recover one column of the dictionary A. It is stated additionally in [QZL$^+$19] that repetitive independent trails lead to a good approximation of the whole dictionary. So if we find multiple different solutions of the optimization problem in (4.4) by using the power method, we are able to get the full dictionary according to [QZL$^+$19]. For this reason we used the power method in order to find a new column of the dictionary that we have not recovered before, until we have found all m atoms of the dictionary.

5.2 Initialization approaches for power method

During the implementation phase, some questions arose concerning the ability to find all columns of the dictionary. The first problem with the algorithm is that we do not know if an atom was already found or not. More specifically we have no knowledge on how to define the threshold such that we can differ between equal and different columns of A, because we do not know the minimal distance between two atoms in the dictionary A. For example, lets assume we already have recovered n columns of the dictionary A and then we find another solution of Problem (4.4), that we will denote in the following by q^*. We need to compare this vector q^* to the n already found column vectors. This new vector q^* could be close but still different to one of the n vector that we have already added to the dictionary before. This could have two reasons then. On the one hand, the q^* and the other vectors could really represent different columns of the dictionary A. On the other hand, it could also be that the vector q^* represents the same atom of A, but that they are different because of some noise in the data or numerical errors. So we do not know how to define the threshold or when two vectors represent the same atom and when different ones. The second problem we recognized was that we do not know where to search for the remaining column vectors or in other words how to initialize the iterate in the power method in order to converge to a column that was not found before. This initialization has a great impact on which atom of A that we find. If we always start with nearly the same initialization we will only recover one column of A. In the paper [QZL$^+$19] no details are given on how to initialize this iterates. So we need to find a way to initialize the iterate such that we recover a new atom that was not found in the iterations before. We therefore came up with different initialization approaches.

Initialization method 1 The first thing, we tried was to take m points that are uniformly distributed over $\mathbb{S}^{d-1}$ as initialization values in the power method. This is the simplest and most intuitive approach. We hoped that the different points will converge to different columns or solutions of Problem (4.4) then.

Initialization method 2 The second approach s to take $K > m$ samples and cluster them to m different clusters. The K initialization are again chosen to be uniformly distributed over the sphere. The centers of these clusters then make up the columns of the dictionary.

Initialization method 3 The last approach was to initialize the iterate of the power method in the orthogonal complement of the columns already added to the dictionary. So as long as we have found less than d columns, we project a randomly chosen vector on the orthogonal complement of the atoms already found. So if we have found $n-1$ vectors as solution by using the power method and then search for the n-th atom, we initialize a vector at random and then calculate the orthogonal complement of the $n-1$ vectors. Afterwards we project the randomly chosen vector on the orthogonal complement and use this as iterate in the following to calculate a solution of Problem (4.4). The remaining $m-d$ iterates are initialized by taking uniformly distributed points over the sphere.

5.3 Algorithm to recover full dictionary

The whole algorithm to recover the full dictionary A is presented in Algorithm 2. The return value is a collection of the different solutions that were recovered by the power method.

Algorithm 2 Dictionary Learning via ℓ_4-norm minimization

Input: data matrix $Y \in \mathbb{R}^{d\times p}$, sparsity level $s \in \mathbb{N}$, initial dictionary $A \in \mathbb{R}^{d\times m}$
  $i = 0$
  $\tilde{A} \in \mathbb{R}^{d\times N}$
  while $i < N$ **do** → not all atoms have been found
    q = power Method(Y, s, A) → independent trials of power method
    if q is different to $\tilde{a}_1, \dots, \tilde{a}_{i-1}$ **then** → atom q has not been recovered before
      $\tilde{a}_i = q$ → add it to the dictionary
      $i \leftarrow i + 1$
    end if
  end while → all atoms are found
Output: the dictionary $\tilde{A} = (\tilde{a}_1, \dots, \tilde{a}_m)$

What has to be mentioned here is that in [QZL+19] the simulation of the test data included the knowledge of the dictionary A, that needed to be recovered, beforehand. Therefore comparisons between the given and fixed dictionary and the recovered dictionary as well as statements, concerning the ability to recover all columns of the dictionary, were made based on these observations.

In the next section some numerical simulations on synthetic and real image data were made to see how Algorithm 2 performs. We also took a look at the importance of Assumption (4.1)

and Assumption (4.2) on the recovery ability and compared the dictionaries of Algorithm 2 and the different initialization approaches to the dictionaries generated by the alternating projection methods.

Chapter 6

Experiments

In order to evaluate the approach presented by [QZL+19], we made some experiments on image and synthetic data. The algorithm implemented extends the code of [PSKK20] by the dictionary learning algorithm in [QZL+19], presented in Chapter 4 and 5. Problem (4.4) was solved, as aforementioned, with the help of the so called power method, described in Algorithm 1. Some methods needed for dictionary learning were implemented as well by translating a *Matlab* code proposed by Karin Schnass to *Python*.

We evaluated the ability to learn the dictionary in two different experiments. First of all we used synthetic data with a known dictionary in order to examine the capability of recovering the dictionary atoms. In the second step, we evaluated the dictionary learning algorithms on image data, where the dictionary was unknown. Here we wanted to take a look at the ability of the algorithm to learn a satisfying and meaningful dictionary, that can for example be used for decomposition of the images or for denoising pictures and compare the dictionaries to the ones of the methods K-SVD and ITKrM.

6.1 Tests on synthetic data

First of all we tested the algorithm by creating some synthetic data. Therefore a specific dictionary A has to be defined or set. Afterwards we created some sparse signals out of this dictionary. With these signals we then learned a dictionary and compared the learned dictionary to the initially set one A.

Dictionary We here used the Dirac-Hadamard matrix as dictionary. This matrix is the concatenation of the identity matrix with a Hadamard matrix. Just for clarification, a Hadamard matrix has only entries in the set $\{1, -1\}$ and the columns are orthogonal to each other, as well as the rows. In order to satisfy the dictionary condition, we also normalize the columns of the generated matrix such that they have unit norm. The Dirac-Hadamard-dictionary fulfills the properties that are required by [QZL+19]: It is a UNTF and the coherence we computed is equal to $\frac{1}{\sqrt{d}}$, which is in our case 0.125 as the dimensions of the dictionary are 64×128. So this Dirac-Hadamard-matrix achieves all the assumptions that were made on the dictionary in the theorem (Assumption (4.2) and Assumption (4.1)).

Signal model We then created N noiseless and s-sparse signals by using the signal model in Table 1 of [Sch15b]. This signal model takes the sparsity level s and a set dictionary A to

generate some perfectly sparse signals. One training signal y is generated in the way that

$$y = Ax \tag{6.1}$$

where $x \in \mathbb{R}^m$ is a sparse coefficient. The vector x is generated by a random permutation p of a sequence c provided with a random set of signs. The condition on the sequence c is that it is monotone decreasing and has unit norm. The sign sequence $\sigma \in \{-1, 1\}^m$ is chosen uniformly at random. As coefficient sequence c we use here $c_i = \frac{1}{\sqrt{s}}$ for $0 \leq i < s$ and $c_i = 0$ for $i > s$. Therefore we can ensure that $\|c\|_2^2 = 1$. The i-th entry of x is then given as

$$x_k = \sigma_k c_{p(k)} \tag{6.2}$$

where $p(k)$ denotes the k-th entry of the permutation of p. We generate a set of training signals y_n and form the training data set as $Y = (y_1, \ldots, y_N)$.

Comparison of learned dictionary with fixed and predefined one As dictionary learning algorithms we used ℓ_4-norm minimization with the generated training signals and compared the learned dictionary to the Dirac-Hadamard matrix fixed in the first step. We compared an atom of the predefined dictionary to the atoms learned and considered it to be recovered if its inner product with one atom of the learned dictionary is bigger than some constant $c \in [0.9, 0.99]$. So for the generated dictionary A and the learned dictionary $\tilde{A}$, the column a_i is recovered if

$$\max_j |\langle a_i, \tilde{a}_j \rangle| \geq c. \tag{6.3}$$

6.1.1 Amount of recovered atoms with respect to iterations needed

The first experiment that was considered in this work is to examine how long it takes to recover all columns of our synthetic dictionary. We therefore generated a Dirac-Hadamard matrix of shape 64×128 and $N = 100000$ sparse signals with the signal model from before and $s = 6$. Additionally, we created K uniformly distributed points on the unit sphere in $\mathbb{R}^d$. As long as not all atoms could be recovered, a new point out of this set of K vectors, that has not been used in the steps before, was chosen and taken as initialization of the iterate in the power method. $K = 2\log(m)m$ points were considered in this experiment. By comparing the solutions found by the power method to the columns of initially set dictionary, we were able to say how fast the algorithm recovers all atoms. For comparison we considered an atom to be recovered if the inner product between a column of the learned dictionary and the atom itself was bigger than some constant c, as described above. Here we set $c = 0.99$. The recovery percentage with respect to the iteration is shown in Figure 6.1 for the Dirac-Hadamard matrix. We can see that the amount of points was enough to recover all atoms of the Dirac-Hadamard matrix. So the algorithm worked for this dictionary, which has low coherence and is a UNTF. Therefore if the assumptions of the theorem hold the algorithm can indeed recover the dictionary A. However, we needed about 550 iterations to recover for $m = 128$ atoms.

Afterwards we wanted to test the ability of the algorithm to recover the columns of the Dirac-Hadamard dictionary, if we disturb or modify a few columns. We therefore added a weighted sum of the columns with index 8 to 64 to the columns 65 to 72 and looked at how long the recovery takes or if we are even able to recover all columns of the modified dictionary. The weight factor was set to $\frac{1}{\sqrt{8}}$. The dictionary was normalized after the modification and $N = 100000$ sparse signals were generated out of this modified dictionary by using the same

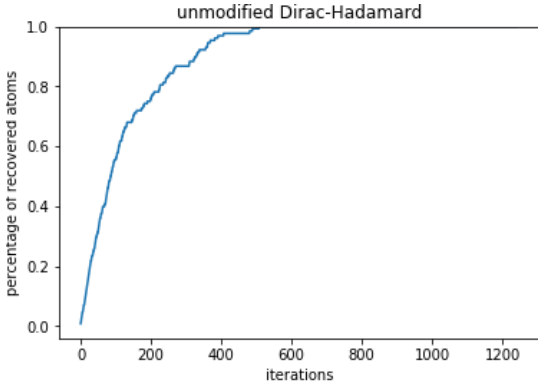


Figure 6.1: Recovery percentage of the atoms of the Dirac-Hadamard dictionary. $2m\log(m)$ uniformly distributed points over the sphere were taken as initialization for $s = 6$ and $c = 0.99$.

signal model as described above or in [Sch15b]. Again we initialized the iterate in the power method by using $K = 10\log(m)m$, so by more than 6000 points. Then we used ℓ_4-norm minimization and tried to recover all dictionary atoms. The recovery percentage can be found in Figure 6.2 for $c = 0.99$.

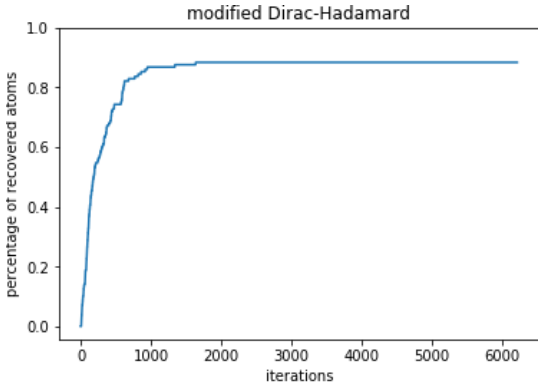
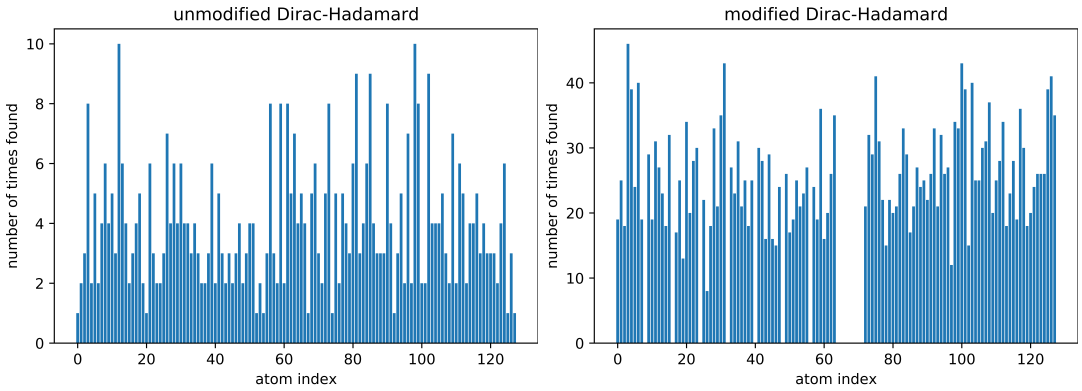


Figure 6.2: Recovery percentage of the atoms of the modified the Dirac-Hadamard dictionary. $10m\log(m)$ uniformly distributed points over the sphere were taken as initialization for $s = 6$ and $c = 0.99$.

Even though we used more points, we were not able to recover all the atoms of the modified dictionary. Only 113 atoms out of $m = 128$ could be found. A reason for this could be that by disturbing these few signals of the modified dictionary by the weighted sum we destroyed the UNTF property. So the modified dictionary is not a unit norm tight frame (UNTF) anymore. Concerning the coherence, the unmodified dictionary achieves a coherence of 0.125 as pointed out before whereas the modified Dirac-Hadamard matrix has a coherence of 0.31. The coherence is more than doubled in the case of the modified dictionary compared to the unmodified Dirac-Hadamard matrix. Therefore the assumptions in (4.1) and (4.2) seem necessary in order to recover all atoms of the dictionary. The question then was which columns could be recovered by this ℓ_4-norm minimization and why some others could not be recovered. For this purpose, we took a look at how often specific columns had been found for the unmodified and the modified dictionary, again by considering $c = 0.99$.
We can see in Figure 6.3(a) where no modifications were made on the Dirac-Hadamard dictionary, we are able to recover all columns at least once. When taking a look at the modified



(a) Dirac-Hadamard dictionary (b) modified Dirac-Hadamard dictionary

Figure 6.3: (a) Recovery amount of atoms of the unmodified the Dirac-Hadamard dictionary. $2m\log(m)$ uniformly distributed points over the sphere were taken as initialization for $s = 6$ and $c = 0.99$. (b) Recovery amount of atoms of the modified the Dirac-Hadamard dictionary. $10m\log(m)$ uniformly distributed points over the sphere were taken as initialization for $s = 6$ and $c = 0.99$.

dictionary, we see that the atoms with index 65 to 72 have not been found at all. But also some columns in the index range from 1 to 64 could not be found either. The reason why this is the case, is because these atoms are less orthogonal meaning they have a higher coherence. This can be seen by taking a look at the Gram-matrix of the modified Dirac-Hadamard-matrix, displayed in Figure 6.4.

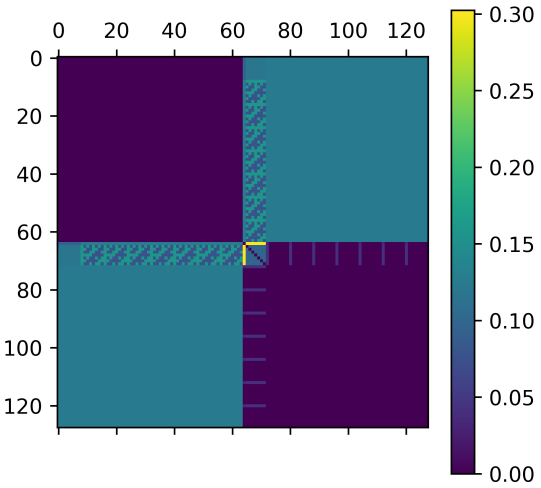
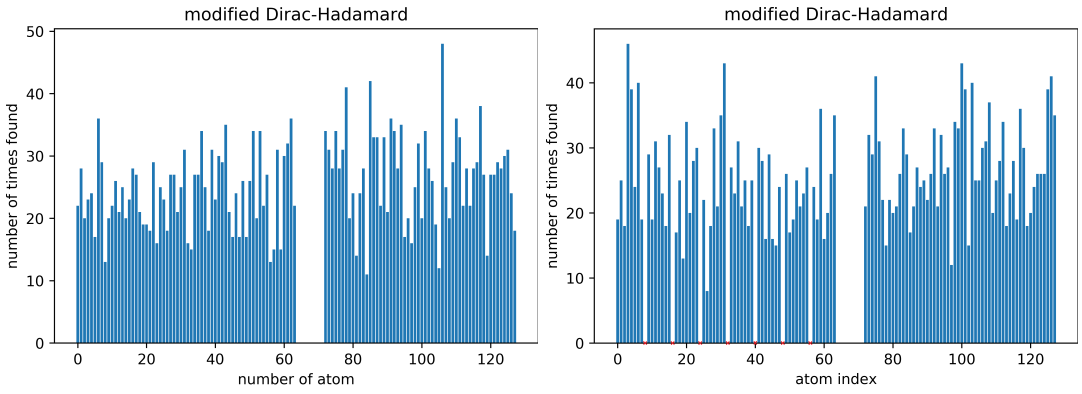


Figure 6.4: Gram-matrix of the modified Dirac-Hadamard dictionary

The Gram-matrix was computed by taking the transposed dictionary and multiplying it with the original one. Afterwards we subtracted the diagonal, because each normalized vector has of course a coherence of 1 with itself. This Gram-matrix shows the coherence of the columns to each other, where brighter colors indicate that the corresponding vectors are more coherent to each other. We clearly see that the atoms 65 to 72 achieve a much higher coherence than the other atoms. But also the atoms in the index range 1 to 64 that could not be recovered either, are a bit more coherent than the rest of the columns. Therefore the algorithm cannot

recover atoms that have a higher coherence. However, if we set the c value down to e.g. 0.95 we could find all columns in the range from index 1 to index 64. The modified atoms at the indices 65 to 72 still could not be recovered, as can be seen in Figure 6.5(a).



(a) $c = 0.95$ (b) $c = 0.99$

Figure 6.5: Recovery amount of atoms of the modified the Dirac-Hadamard dictionary. $10m\log(m)$ uniformly distributed points over the sphere were taken as initialization for $s = 6$ and (a)$c = 0.95$ (b) $c = 0.99$.

Taking a closer look at the vectors at the index positions 65 to 72, it is to say that they are close to the left singular vector of the modified dictionary. To see this we computed the absolute values of the inner products between the atoms 65 to 72 and the left singular vector of the modified dictionary. This absolute value is a measurement for the coherence between the left singular vector and the modified atoms. The corresponding plot of this inner products can be found in Figure 6.6.

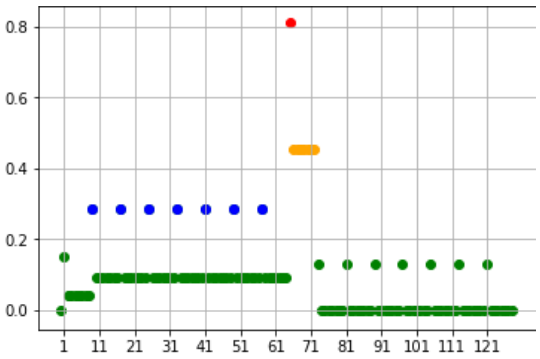
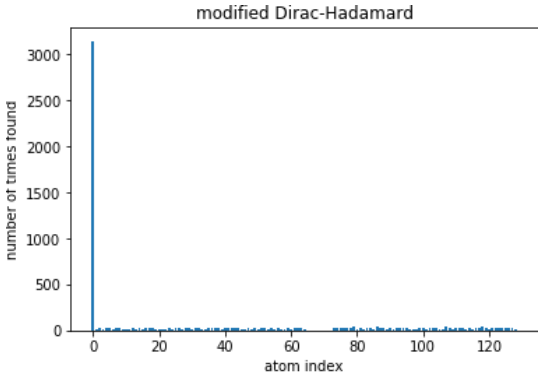


Figure 6.6: Inner products of the atoms of the modified Dirac-Hadamard dictionary with the left singular vector

The atoms at position 65 to 72, displayed in red and orange, are made up by a large component in the direction of the left singular vector. Especially the coherence between the red marked atom 65 and the left singular vector is very high, but also the coherence of the atoms 66 to 72 is much higher as the coherence of any other atom with the left singular vector. The atoms in the fist half which achieve a coherence of approximately 0.3, displayed here in blue, are exactly the ones that could not be found in the first experiment of this section, where we considered $10m\log(m)$ uniformly distributed points over the sphere as initialization for the ℓ_4-norm minimization for $s = 6$ and $c = 0.99$.

6.1.2 Recovery of the left singular vector

The next experiment considered was to see how often we recover a vector near the left singular vector by using ℓ_4-norm minimization. We therefore added the left singular vector at index zero of the modified Dirac-Hadamard dictionary and again compared the vectors found by the power method to the modified Dirac-Hadamard dictionary and additionally the left singular vector. We set $c = 0.95$ and $10m\log(m)$ uniformly distributed points as initialization values. We can see that vectors near the left singular vector were recovered by far the most often, as illustrated in Figure 6.7(a).



(a) $c = 0.95$

Figure 6.7: Recovery amount of atoms of the modified the Dirac-Hadamard dictionary. $10m\log(m)$ uniformly distributed points over the sphere were taken as initialization for $s = 6$, $c = 0.95$

So in conclusion, if the dictionary fits perfectly the assumptions made, we are able to recover the atoms of the dictionary by using ℓ_4-norm minimization . But if we disturb the dictionary in some atoms, we cannot recover the more coherent atoms anymore, even if the data consists of perfectly sparse and noiseless signals. This is not promising when coming to real image data in the next subsection. When considering image data, the dictionaries of the data will not be a UNTF and also the coherence is much higher in most of the cases. Therefore the chance of recovering all columns of the dictionary for images may not be that high.

6.2 Tests on image data and comparison with K-SVD and ITKrM

In the second setting, as mentioned before, we used image data and compared the results of the dictionary learning methods K-SVD, ITKrM and ℓ_4-norm minimization.

Data Here we used the famous computer vision image *Barbara*, displayed in Figure 6.8. The image was rescaled to a size of 256×256 pixels. In order to work with images, first we had to calculate the patches out of this image, which represent the different signals in the matrix Y. On these patches we learned the dictionary and the sparse coefficients by using alternating projection methods or ℓ_4-norm minimization. In this context we also want to specify what a patch is. The picture or image that is given represents a data matrix, as every pixel stands for a given color and number. This matrix is then divided into smaller submatrices, the patches, of size 8×8. This patch matrix was then flattened to a vector and represents one signal or

column of the data matrix Y. For the dictionary learning methods we used all available 62001 patches generated out of the *Barbara* image. In order to generate a meaningful dictionary we also have to project the patches onto the orthogonal complement of the so called flat-atom, defined as $(\frac{1}{2}, \ldots, \frac{1}{d}) \in \mathbb{R}^d$. This flat-atom corresponds to a low dimensional subspace and contains most of the energy of the signal. Otherwise, if we would not project the patches on the orthogonal complement of this flat-atom, this would be the dominant factor that we would recover out of every patch.

Figure 6.8: *Barbara* image

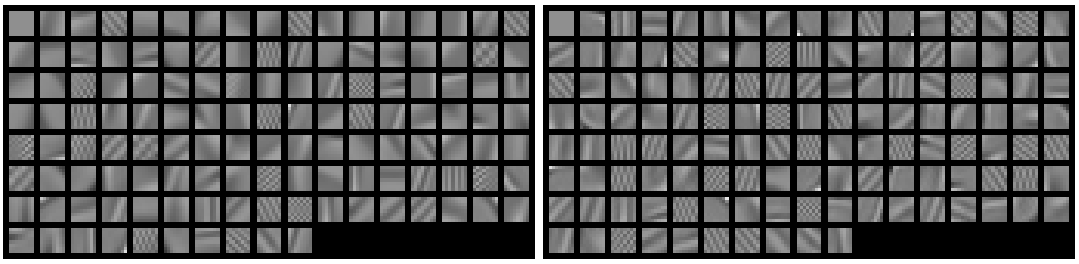
Initialization In the following, if not specified otherwise, the sparsity level was set to $s = 6$. We chose the size of the dictionary to be $(64, 128)$. For the visualization of the dictionary and the recovery of the image patches we added the flat-atom to the dictionary. The maximum number of iterations in K-SVD and ITKrM was set to 100, the maximum number of iterations in the power method set to 20.

Comparison/Error To evaluate the success of recovery we used the peak signal-to-noise ratio (PSNR) between the original *Barbara* image B and the reconstructed image $\tilde{B}$. In order to find the sparse coefficients we used OMP for $s = 6$. The reconstructed image was then generated by the multiplication of the learned dictionary and these recovered sparse coefficients. For two images of the same size $d_1 \times d_2$ and pixel values in $[0, 255]$ PSNR is defined as

$$\text{PSNR in dB} = 10 \log_{10} \left(\frac{255^2}{\frac{1}{d_1 d_2} \sum_{i,j} (B_{i,j} - \tilde{B}_{i,j})^2} \right). \tag{6.4}$$

The higher the PSNR value is, the better is the reconstructed image. To compute the PSNR values we re-scaled all images to a shape of $(d1, d2) = (256, 256)$ pixels.

The dictionaries learned on the *Barbara* image by using K-SVD and ITKrM are portrayed in Figure 6.9. The coherence of the dictionary that was recovered by the ITKrM algorithm was 0.7377 and for K-SVD we achieved a coherence of 0.9590 for $s = 6$. So there are probably at least two similar atoms in each of our dictionaries.
Then we recovered the dictionaries by using ℓ_4-norm minimization together with the presented initialization approaches for the power method.



(a) K-SVD (b) ITKrM

Figure 6.9: Dictionaries learned on the image for sparsity level $s = 6$ with initial dictionary size $d = 64$, $m = 128$ by algorithm (a) K-SVD and (b) ITKrM.

6.2.1 Initialization method 1: Initialization with uniformly distributed points

In the first setting, we took m points that are uniformly distributed over $\mathbb{S}^{d-1}$ and initialized the iterate in the power method by these points. The learned dictionary by using this initialization approach can be found in Figure 6.10

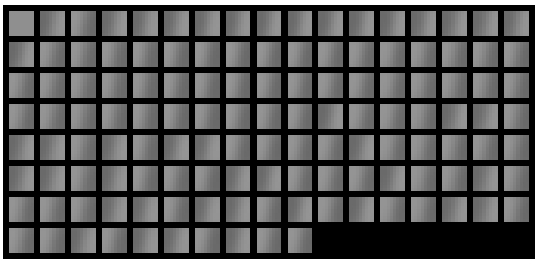


Figure 6.10: Dictionary learned on the image for sparsity level $s = 6$ with initial dictionary size $d = 64$, $m = 128$ by using ℓ_4-norm minimization and Initialization method 1.

When comparing the dictionary learned on the Barbara image to the ones in Figure 6.9, we can see that we could not recover a meaningful dictionary for the first approach on image data. All atoms look similar to each other and have again a large inner product with the left singular vector. When taking a look at the tablecloth in the Barbara image and the atom at index position $(2, 16)$ in the dictionary learned by K-SVD or the atom with index $(1, 14)$ for ITKrM, we cannot find any similar looking atom in the dictionary we get by using Initialization method 1 of ℓ_4-norm minimization. Therefore we will not be able to recover e.g. the tablecloth with its specific design by using this dictionary.

6.2.2 Initialization method 2: Clustering of atoms

In the second approach we took $K = 4m \log m$ uniformly distributed points over the sphere for $m = 128$ and clustered them to m different clusters. The clustering here was done by using the machine learning library *sklearn* with the algorithm *Kmeans*. This algorithm takes the number of different clusters m, the computed samples Z, with $|Z| = K$, and divides the points into the clusters C by choosing the respective cluster which minimizes the inertia. The inertia is defined in the following way

$$\sum_{i=0}^{N} \min_{\mu_j \in C} \|z_i - \mu_j\|^2.$$

The value μ_j describes the mean of the points that are already in cluster j. In the initial step m points are chosen to build the different cluster centers [PVG+11] and the other points are added as described above. What has to be considered here is that an atom in the dictionary can easily be replaced by its negative by a simple sign change in the sparse coefficients. The amount of nonzero entries stays the same. Therefore in order to cluster these vectors into m different clusters, that build our dictionary atoms, we need to ensure that opposite vectors on the sphere are in the same cluster. In our implementation this was done by projecting each vector in the same hemisphere and performing the clustering algorithm afterwards. The learned dictionary can be found in Figure 6.11.

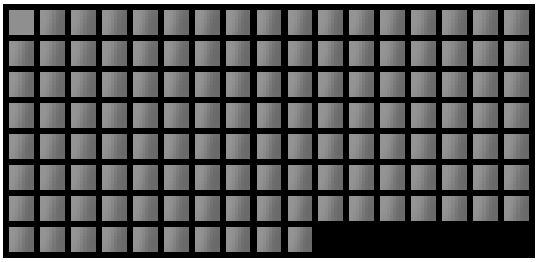


Figure 6.11: Dictionary learned on the image for sparsity level $s = 6$ with initial dictionary size $d = 64$, $m = 128$ by using ℓ_4-norm minimization and Initialization method 2. $4m \log m$ points were considered and the solutions were clustered by *Kmeans*.

Concerning this dictionary, it is similar to the one in the first approach. Again there is no atom in the dictionary that looks like one of the atoms $(2, 16)$ of K-SVD or $(1, 14)$ of ITKrM. So the atom which is probably used for the decomposition of the tablecloth in the *Barbara* image, is again not a part of the dictionary in Figure 6.11.

6.2.3 Initialization method 3: Initialization with orthogonal projection

The last approach was to project random vectors on the orthogonal complement of the columns already added to the dictionary for the first d atoms and initialize the others uniformly over the sphere. The dictionary we could learn via this initialization approach is visualized in Figure 6.12.

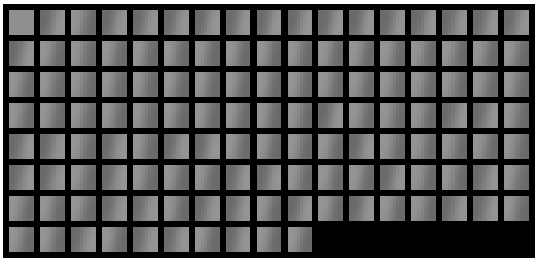


Figure 6.12: Dictionary learned on the image for sparsity level $s = 6$ with initial dictionary size $d = 64$, $m = 128$ by using ℓ_4-norm minimization and Initialization method 3.

Also in the last case we are not able to recover a dictionary with whom we can approximate the data properly or the atom used for the decomposition of the tablecloth.

6.2.4 Approximation plots

To see how our algorithm performed, we took a look at the corresponding approximation plots. For that we generated the sparse coefficients of the given dictionaries by using OMP. Then we calculated the patches of the image approximation by multiplying the dictionaries with their sparse coefficients. In order to display the image out of this approximation we had to transform the patches into submatrices of the output image. The approximation plots of K-SVD and ITKrM are shown in Figure 6.13(a,b). As the 3 different initialization methods of ℓ_4-norm minimization recovered nearly the same dictionary we considered here only Initialization method 1 for the image approximation. The approximation plot we could generate is displayed in Figure 6.13(c).

(a) K-SVD (b) ITKrM

(c) Initialization method 1

Figure 6.13: Approximation plots by using (a) K-SVD (b)ITKrM (c) ℓ_4-norm minimization and Initialization method 1.

We then used these approximation plots and evaluated the PSNR value between the original *Barbara* image and each approximation. The results can be found in Table 6.1.
We can see that ITKrM and K-SVD could achieve really good measurements concerning the PSNR value and also by comparing the approximation plots itself to the original image, we cannot see any big difference. ℓ_4-norm minimization and Initialization method 1 did not lead

Dictionary learning algorithm	PSNR
ITKrM	83.92 dB
K-SVD	83.61 dB
ℓ_4-norm minimization Initialization method 1	25.47 dB

Table 6.1: PSNR values of the approximation plots in Figure 6.13 with the original image in Figure 6.8

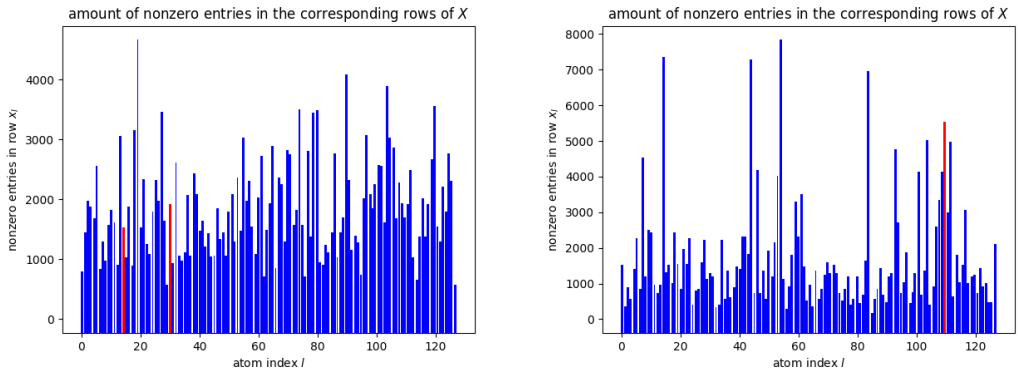
to such good approximation of the image, the reconstructed image is much more blurred compared to the approximation plots of K-SVD and ITKrM. Also the PSNR value out of this approximation is lower. The reason why ℓ_4-norm minimization does not lead to such good results as the alternating projection methods to could be that the images do not fulfill the assumptions that were made in [QZL$^+$19]. Neither the UNTF constraint, the low coherence nor most likely the Bernoulli-Gaussian distribution of the sparse coefficients are met in this setting. Therefore alternating projection methods are properly the better choice for dictionary learning applications on images.

6.2.5 Usage of the left singular vector for data decomposition

As mentioned before all recovered atoms by ℓ_4-norm minimization are near the left singular vector or its negation respectively. Therefore the next experiment that we took into account was to take a look at the importance of this left singular vector for the decomposition of the data. We used here the dictionaries of K-SVD and ITKrM and recovered the sparse coefficients X out of theses dictionaries by using the OMP algorithm for each dictionary. Afterwards, the atoms of the dictionaries that are near the left singular vector were evaluated. This was done by computing the absolute values of inner products of the left singular vector and each atom. For K-SVD one atom had a inner product of more than 0.95 with the left singular vector, whereas two atoms of the ITKrM dictionary could achieve an inner product of about 0.86 with the left singular vector. So these atoms have a large component in the direction of the left singular vector. For an atom at index i the corresponding i-th row in X defines how much impact this atom has. So to measure the importance of the atoms, we counted the amount of nonzero entries and the ℓ_1-norm of the corresponding rows in the coefficient matrix X. The atoms in the corresponding dictionary that were close to the left singular vector are displayed in red.
We can see in Figure 6.14 and Figure 6.15 that the atoms near the left singular vector play an important role concerning the amount of usage and the sum of the absolute coefficient values. This vector is used very often in the case of K-SVD and ITKrM. So if we come back to ℓ_4-norm minimization, the left singular vector was close to all atoms that we recovered by this algorithm. More specifically, the absolute value of the inner products between the left singular vector and any atom recovered by the ℓ_4-norm minimization were within the range $(0.95, 1]$. Thus we can say that we could at least find a very important atom for the decomposition or approximation of the data by ℓ_4-norm minimization.

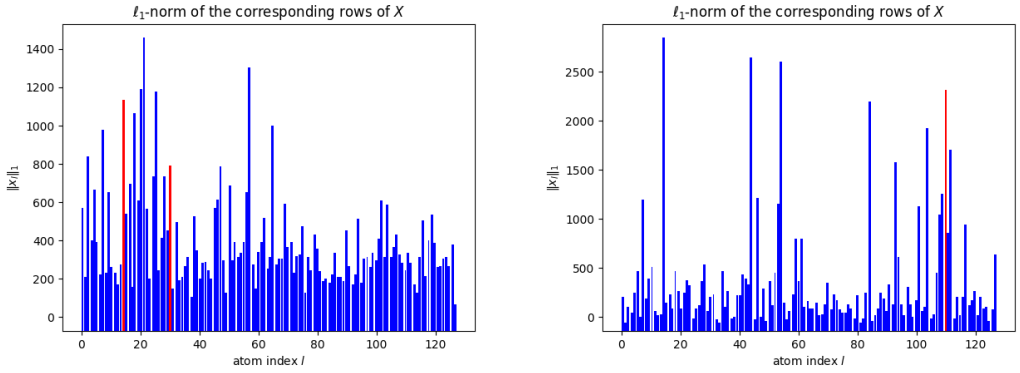
6.2.6 Removal of multiple low-rank components

The last experiment that was considered here is based on the approach presented by [NS18]. Here we projected the patches on the orthogonal complement of multiple low-rank components. These low-rank components again contain lots of signal energy. The reason why we consider



(a) K-SVD (b) ITKrM

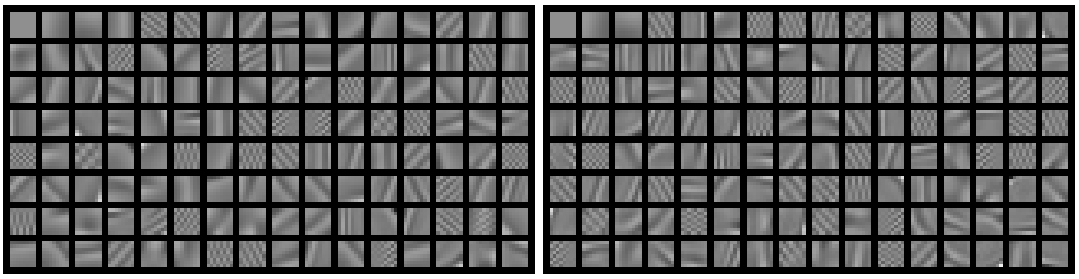
Figure 6.14: Amount of times each atom is used by counting the nonzero entries in the sparse coefficients X. The red vectors are close to the left singular vector. (a) K-SVD atom usage (b) ITKrM atom usage.



(a) K-SVD (b) ITKrM

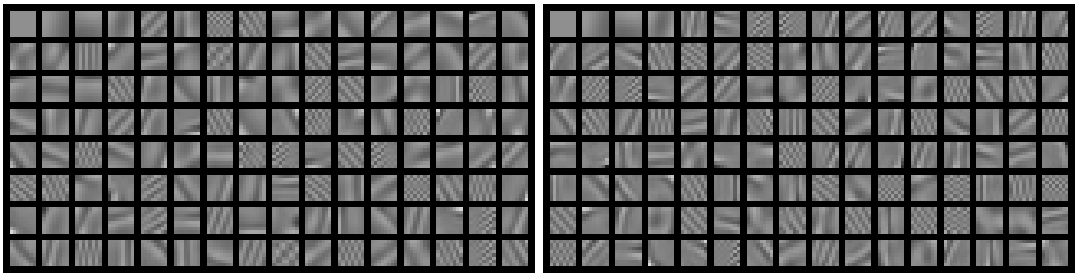
Figure 6.15: ℓ_1-norm of the corresponding rows in X of each dictionary atom . The red vectors are close to the left singular vector. (a) sum of absolute row values per K-SVD atom (b) sum of absolute row values per ITKrM atom.

this experiment here is because natural signals like images normally are not perfectly sparse. This means they can be modeled as orthogonal sum of a low-rank and a sparse component [NS17b]. So if the signals of the data are not perfectly sparse, it could be that they are distorted towards these low-rank components. We used here the L largest singular values of the patches as low-rank components and projected the patches out of the *Barbara* image onto the orthogonal complement of these vectors. The calculation of this left singular vectors was done by SVD. The left singular vector corresponding to the largest singular value is nearly equal to the flat-atom that we considered in the last experiments. The size of the of dictionary was again set to $d = 64$ and $m = 128 - L$. The dictionary we could learn by taking this modified patches as input data as well as the sparsity level $s = 6$ can be found in Figure 6.16 for 3 left singular vector atoms and in Figure 6.17 for 4 left singular vectors each for K-SVD and ITKrM.
The dictionaries that could be recovered by ℓ_4-norm minimization and Initialization method



(a) K-SVD and 3 left singular vectors (b) ITKrM and 3 left singular vectors

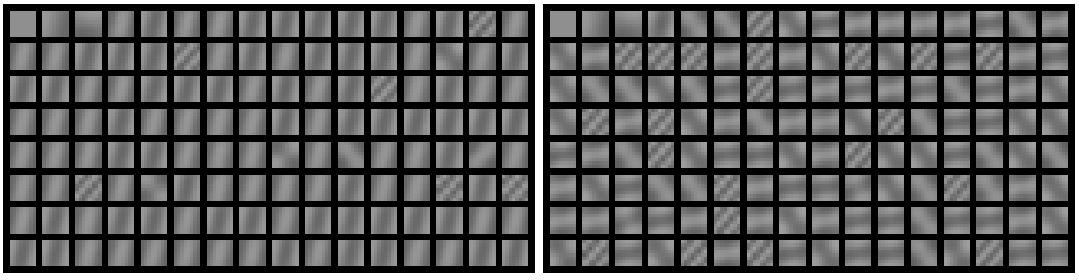
Figure 6.16: Dictionaries learned on the image for sparsity level $s = 4$ with initial dictionary size $d = 64$, $m = 128$ by algorithm (a) K-SVD and (b) ITKrM. The patches, the dictionaries were learned on, were projecting on the orthogonal complement of the largest three left singular vectors of the image patches.



(a) K-SVD and 4 left singular vectors (b) ITKrM and 4 left singular vectors

Figure 6.17: Dictionaries learned on the image for sparsity level $s = 4$ with initial dictionary size $d = 64$, $m = 128$ by algorithm (a) K-SVD and (b) ITKrM. The patches, the dictionaries were learned on, were projecting on the orthogonal complement of the largest four left singular vectors of the image patches.

1 for $L = 3$ and $L = 4$ can be found in Figure 6.18.



(a) ℓ_4-minimization and 3 left singular vectors (b) ℓ_4-minimization and 4 left singular vectors

Figure 6.18: Dictionaries learned on the image for sparsity level $s = 4$ with initial dictionary size $d = 64$, $m = 128$ by using ℓ_4-norm minimization and Initialization method 1. The patches, the dictionaries were learned on, were projecting on the orthogonal complement of the largest four left singular vectors of the image patches.

As we can see if we project the patches of the image onto the orthogonal complement of more low-rank components, we are able to recover atoms which are different to the ones in e.g.

Figure 6.10. We can also make approximation plots out of these dictionaries. We here took the dictionaries of K-SVD, ITKrM and ℓ_4-norm minimization for $L = 4$ low-rank components into account. The approximation plots can be seen in Figure 6.19.

(a) K-SVD

(b) ITKrM

(c) ℓ_4-norm minimization and Initialization method 1

Figure 6.19: Approximation plots by using (a) K-SVD (b)ITKrM (c) ℓ_4-norm minimization and Initialization method 1. $L = 4$ low-rank components were subtracted from the patches.

Again we computed the PSNR values between the original *Barbara* image and each approximation. The results can be found in Table 6.2.

Dictionary learning algorithm	PSNR
ITKrM	82.89 dB
K-SVD	82.91 dB
ℓ_4-norm minimization Initialization method 1	76.18 dB

Table 6.2: PSNR values of the approximation plots in Figure 6.19 with the original image in Figure 6.8. $L = 4$ low-rank components were subtracted from the patches.

The approximation we could achieve by this ℓ_4-norm minimization dictionary is quite good compared to the ones in Figure 6.13(c). However the PSNR value of ℓ_4-norm minimization is still lower than those of K-SVD and ITKrM in Table 6.1 and in Table 6.2. Also the plot in Figure 6.19(c) is more blurred than the one generated by K-SVD or ITKrM.

Chapter 7

Conclusion

Dictionary learning deals with finding an dictionary A and sparse coefficients X such that we can decompose a given set of signals with a small approximation error. The main paper that is studied in this thesis, [QZL+19], formulates the dictionary learning problem in the overcomplete case as a maximization problem of the ℓ_4-norm under certain constraints. It has been shown that every solution of this problem is close to an atom of the dictionary A and that there are no spurious minimizers. In order to assess the practical relevance of the algorithm, the proposed approach was implemented. We made use of the so called power method, an iterative algorithm which finds the minimum value of a function over a convex set to recover the columns of the dictionary. In the algorithm, we faced the problem that we could not guarantee that we have found all columns of the dictionary A because we had no knowledge on how to initialize the vectors such that a new column of the dictionary A could be recovered. Therefore we developed different initialization approaches to overcome the recognized problems: The first one was to take m uniformly distributed vectors over the sphere as initialization in the method and hope that they may converge to the m different atoms of the dictionary A then. The second thing we tried was clustering, where we generated even more points and clustered them into m different clusters. Afterwards, we took the cluster centers as and added them to the dictionary A. The next approach was orthogonal projection, where we projected the a random vector on the orthogonal complement of the already recovered atoms for the first d components and took this projected vector as initialization value. To recover the remaining $m - d$ atoms we used uniformly distributed points over the sphere.
The algorithm was tested then on synthetic data as well as on image data by using the Barbara image. Considering synthetic data, first we considered a dictionary, which fulfilled all the assumptions made in [QZL+19], and generated some perfectly sparse signals. We were able to recover all column of the dictionary at least one time, but we needed lots of iterations. By changing the dictionary such that the assumptions were met anymore, we were not even able to recover all atoms. On image data we compared the dictionaries of the alternating projection methods K-SVD and ITKrM to the three initialization approaches of ℓ_4-norm minimization. We saw that the dictionaries recovered by ℓ_4-norm minimization could not recover all atoms of the dictionary. We only found vectors, that were close to the left singular vector. A reason could be that on image data the assumptions or constraints on the dictionary do not hold. So the algorithm is not robust or stable if not all the assumptions are met perfectly and it cannot keep up with the alternating projection methods. Only by subtracting more low-rank components we could recover a better dictionary by using ℓ_4-norm minimization and generate a better approximation plot of the image. Nevertheless, the results of K-SVD and ITKrM were still better and therefore it is more advisable to use alternating

projection methods for dictionary learning on images.

Bibliography

[AEB06] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54:4311 – 4322, 2006.

[BE08] O. Bryt and M. Elad. Compression of facial images using the k-svd algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, 2008.

[EA06] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.

[EAH99] K. Engan, S. Aase, and J. Husoy. Method of optimal directions for frame design. volume 5, pages 2443 – 2446 vol.5, 02 1999.

[LBM11] K. Labusch, E. Barth, and T. Martinetz. Robust and fast learning of sparse codes with stochastic gradient descent. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):1048–1060, 2011.

[MBP12] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791–804, 2012.

[MZ93] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.

[NS17a] V. Naumova and K. Schnass. Dictionary learning from incomplete data. *arXiv*, 2017.

[NS17b] Valeriya Naumova and Karin Schnass. Dictionary learning from incomplete data for efficient image restoration. pages 1425–1429, 2017.

[NS18] V. Naumova and K. Schnass. Fast dictionary learning from incomplete data. *EURASIP Journal on Advances in Signal Processing*, December 2018.

[PSKK20] M. Pali, T. Schaeffter, C. Kolbitsch, and A. Kofler. Adaptive sparsity level and dictionary size estimation for image reconstruction in accelerated 2d radial cine mri. *Medical Physics*, 2020.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[QZL+19] Q. Qu, Y. Zhai, X. Li, Y. Zhang, and Z. Zhu. Analysis of the optimization landscapes for overcomplete representation learning. *CoRR*, abs/1912.02427, 12 2019.

[RBE10] R. Rubinstein, A. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

[SBZJ14] M. Sadeghi, M. Babaie-Zadeh, and C. Jutten. Learning overcomplete dictionaries based on atom-by-atom updating. *IEEE Transactions on Signal Processing*, 62(4):883–891, 2014.

[Sch15a] K. Schnass. Convergence radius and sample complexity of itkm algorithms for dictionary learning. *Applied and Computational Harmonic Analysis*, 45, 2015.

[Sch15b] K. Schnass. Local identification of overcomplete dictionaries. *J. Mach. Learn. Res.*, 16(1):1211–1242, January 2015.

[SQW17] J. Sun, Q. Qu, and J. Wright. Complete dictionary recovery over the sphere i: Overview and the geometric picture. *IEEE Transactions on Information Theory*, 63(2):853–884, 2017.

[TW10] J. Tropp and S. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6):948–958, 2010.