University of Innsbruck

Bachelor Thesis

Hard Thresholding Pursuit for Sparse Approximation

Eva M. Höck

supervised by Karin Schnass

November, 2016

Contents

1 Introduction

In this thesis we examine the Hard Thresholding Pursuit (HTP) algorithm for sparse approximation. The first chapter presents an introduction to sparse approximation and related topics such as different sparse approximation algorithms and dictionaries.

1.1 Sparse approximation

In research fields such as signal processing, sparse approximation is the problem of obtaining the best approximation of a signal while only using few elements of a redundant dictionary to construct the approximation.
To begin with, we have to introduce a few definitions:

Definition 1.1 (Dictionary)**.**
Let d and K be positive integers with $d \leq K$. A matrix $\Phi \in \mathbb{R}^{d \times K}$ is called a *dictionary matrix* or *dictionary* in $\mathbb{R}^d$ if its columns $\Phi_j,\ 1 \leq j \leq K$, satisfy

$$\forall\ 1 \leq j \leq K \colon \|\Phi_j\|_2 = 1.$$

The columns Φ_j of a dictionary Φ are called *atoms*.
Let Φ be a dictionary in $\mathbb{R}^d$. If the atoms $(\Phi_j)_{1 \leq j \leq K}$ are linearly dependent, Φ is called a *redundant* dictionary.

In Chapter 1.3, different types and examples of dictionaries are presented.
The following definition helps us to count the non-zero entries of a column vector and will be useful in defining sparsity.

Definition 1.2 (Support and $\|\cdot\|_0$-function)**.**
Let $x \in \mathbb{R}^K$ be a vector. The *support* of x is defined as

$$\operatorname{supp}(x) := \{j \mid x_j \neq 0\} \subset \{1, \ldots, K\}.$$

The $\|\cdot\|_0$-*function* is defined as

$$\|\cdot\|_0 \colon \mathbb{R}^K \to \{1, \ldots, K\} \colon x \mapsto \|x\|_0 := |\operatorname{supp}(x)|.$$

Note that the $\|\cdot\|_0$-function is not homogeneous and therefore does not satisfy the definition of a norm.

The next definition introduces sparsity. This concept describes how many elements of a dictionary are needed to construct a signal.

Definition 1.3 (Sparsity)**.**
Let $\Phi \in \mathbb{R}^{d \times K}$ be a dictionary matrix, $x \in \mathbb{R}^K$, and $s << d$.
Then $y := \Phi x$ is called *s-sparse in* $\Phi :\Leftrightarrow \|x\|_0 \leq s$.

With the help of this terminology, we can formulate the following two problems:

- Exact sparse recovery:
  Given the dictionary matrix $\Phi \in \mathbb{R}^{d \times K}$, sparsity level $s \in \mathbb{N}$, and $y \in \mathbb{R}^d$ that is s-sparse in Φ, we want to find $x \in \mathbb{R}^K$ with $\|x\|_0 \leq s$ such that $y = \Phi x$.

- Sparse approximation:
  Given the dictionary matrix $\Phi \in \mathbb{R}^{d \times K}$, sparsity level $s \in \mathbb{N}$, and $y \in \mathbb{R}^d$, the objective is to find $x \in \mathbb{R}^K$ with $\|x\|_0 \leq s$ such that $\|y - \Phi x\|_2$ is minimal.

The difference between the two scenarios is that in exact sparse recovery we assume the existence of an s-sparse representation of y in the dictionary, whereas in sparse approximation we want to get as close as possible to such an s-sparse representation. The former is a special case of the latter problem and therefore sometimes both are referred to as sparse approximation.
In both settings we call $\mathbb{R}^d$ the *signal space*. In many cases the atoms form a redundant system that spans the whole signal space. In Chapter 3 for example, we encounter a dictionary that consists of the union of two orthonormal bases of the signal space.
There are different approaches to perform sparse approximation. The main algorithm families are the matching pursuit (MP) family (e. g. [6]), which apply greedy methods, the basis pursuit (BP) family (e. g. [3]), which tackle the problem with convex relaxation, and the thresholding algorithms (e. g. [2]). For an overview of these methods see [5, 13].
This thesis examines the Hard Thresholding Pursuit (HTP) algorithm, which was presented in 2011 by Simon Foucart. This method combines certain components of MP and thresholding.
Having introduced a few definitions and the problems we want to solve, we are going to look at different sparse approximation algorithms in the following section.

1.2 Algorithms for sparse approximation

The following algorithms for sparse approximation require the input of a dictionary matrix $\Phi \in \mathbb{R}^{d \times K}$, a signal $y \in \mathbb{R}^d$, and a sparsity level $s \in \mathbb{N}$ such that $s \ll d$. The algorithms then try to compute a best s-sparse approximation Φx of the signal y.
A crucial point for this objective is how to decide what the support of x should be. In the following we are going to look at three algorithms that employ different methods for this task: the OMP, the IHT, and the HTP algorithm.
First of all we have to introduce a new notation:

Definition 1.4.
Let $A \in \mathbb{R}^{d \times K}$ be a matrix and $I = \{j_1, \ldots, j_n\} \subset \{1, \ldots, K\}$ an index set with $j_1 < \ldots < j_n$.
We define $A_I \in \mathbb{R}^{d \times n}$ as the matrix formed by the columns of A with indices in I, i. e.

$$A_I := [A_{j_1} \ldots A_{j_n}].$$

Some of the algorithms include a linear-least-squares approximation which can be realized using the pseudoinverse:

Definition 1.5 (Pseudoinverse)**.**
Let $\Phi \in \mathbb{R}^{d \times K}$ be a matrix. For $y \in \mathbb{R}^d$ consider the standard linear approximation problem

$$\text{find } x \in \mathbb{R}^K \colon \|\Phi x - y\|_2 \text{ is minimal.} \tag{1}$$

The *pseudoinverse* of Φ is the matrix $\Phi^+ \in \mathbb{R}^{K \times d}$ that satisfies

$$\forall\, y \in \mathbb{R}^d \colon x := \Phi^+ y \text{ solves (1) and } \|x\|_2 = \min\{\|z\|_2 \mid z \in \mathbb{R}^K \text{ solves (1)}\}.$$

Notice that following this definition there is a unique pseudoinverse for all matrices.
Another interpretation of the pseudoinverse is its use in performing a projection onto the linear span of several atoms: Let $I \subset \{1, \ldots, K\}$ be an index set. Then the orthogonal projection operator $P(\Phi_I)$ onto the linear span of the atoms $\Phi_i, i \in I$, satisfies $P(\Phi_I) = \Phi_I \Phi_I^+$.

One way to calculate the pseudoinverse of a matrix uses the singular value decomposition of Φ:

Remark 1.6 (Calculation of the pseudoinverse).
Let $\Phi = U\Sigma V^\top$ be the singular value decomposition of $\Phi \in \mathbb{R}^{d\times K}$ with orthogonal matrices $U \in \mathbb{R}^{d\times d}, V \in \mathbb{R}^{K\times K}$ and rectangular diagonal matrix $\Sigma \in \mathbb{R}^{d\times K}$ containing the singular values of Φ. The pseudoinverse of Φ has the form $\Phi^+ = U\Sigma^+V^\top$ and Σ^+ is a rectangular diagonal matrix containing the inverse non-zero singular values of Φ.

Orthogonal Matching Pursuit

The Orthogonal Matching Pursuit algorithm (OMP) (see [8]) belongs to the MP family and builds the support incrementally:

Algorithm 1.1 (Orthogonal Matching Pursuit)**.**
Input: $\Phi \in \mathbb{R}^{d\times K},\ y \in \mathbb{R}^d,\ s \in \mathbb{N}$
Initialization: $I^0 = \emptyset,\ x^0 = 0,\ n = 0$
Iteration: repeat until $n = s$:

$$
\begin{aligned}
n &:= n + 1 \\
i_n &:= \operatorname*{argmax}_{1\le i\le K} |\langle y - \Phi x^{n-1}, \Phi_i\rangle| \\
I^n &:= I^{n-1} \cup \{i_n\} \\
x_i^n &:= \begin{cases} (\Phi_{I^n}^+ y)_i & \text{if } i \in I^n \\ 0 & \text{if } i \notin I^n \end{cases}
\end{aligned}
$$

Output: $x := x^n$

A discussion of the OMP algorithm and its variants is presented in [5, 13].

Iterative Hard Thresholding

The Iterative Hard Thresholding algorithm (IHT) (see [2]) pursues a different strategy by choosing s indices for the support at once, checking if that choice is good enough and choosing a different support set if it is not.
Before describing the algorithm itself, we have to define the hard thresholding operator:

Definition 1.7 (Hard thresholding operator)**.**
Let s be a positive integer. For $x \in \mathbb{R}^K$ we define $L_s(x)$ as the index set of the s largest (in modulus) entries of x.
The *hard thresholding operator* $H_s \colon \mathbb{R}^K \to \mathbb{R}^K \colon x \mapsto H_s(x)$ is defined by

$$H_s(x)_i := \begin{cases} x_i & \text{if } i \in L_s(x) \\ 0 & \text{otherwise} \end{cases}.$$

The IHT algorithm employs this operator as follows:

Algorithm 1.2 (Iterative Hard Thresholding)**.**
Input: $\Phi \in \mathbb{R}^{d \times K},\ y \in \mathbb{R}^d,\ s \in \mathbb{N}$
Initialization: $x^0 = 0,\ n = 0$
Iteration: repeat until a stopping criterion is met:

$$n := n + 1$$
$$x^n := H_s(x^{n-1} + \Phi^\top (y - \Phi x^{n-1}))$$

Output: $x := x^n$

The stopping criterion can be a maximum number of iterations and/or a residual error tolerance.

Hard Thresholding Pursuit

The Hard Thresholding Pursuit algorithm (HTP) (see [4]) uses the method of choosing the support from IHT and performs a linear approximation with these atoms using the pseudoinverse, which we have already seen in the OMP algorithm.

Algorithm 1.3 (Hard Thresholding Pursuit)**.**
Input: $\Phi \in \mathbb{R}^{d\times K},\ y \in \mathbb{R}^d,\ s \in \mathbb{N}$
Initialization: $x^0 = 0,\ n = 0$
Iteration: repeat until a stopping criterion is met:

$$n := n + 1$$
$$I^n := L_s(x^{n-1} + \Phi^\top(y - \Phi x^{n-1}))$$
$$x_i^n := \begin{cases} (\Phi_{I^n}^+ y)_i & \text{if } i \in I^n \\ 0 & \text{if } i \notin I^n \end{cases}$$

Output: $x := x^n$

Possible stopping criteria are, as for the IHT, a maximum number of iterations and/or a residual error tolerance. Additionally, a sensible criterion is $I^{n+1} = I^n$, because then the algorithm reaches constancy. In this thesis we use the last criterion in combination with the maximum number of iterations being s.
Unlike OMP, which is widely used in sparse approximation, the HTP algorithm is not common in this field yet because of its origins in compressive sensing. This discipline is introduced very briefly in the next section.

Compressive sensing

Originally, the HTP algorithm was designed for compressive sensing, which is similar to sparse approximation. These two fields of research address nearly the same problem, from differing viewpoints and with differing terminologies.
In compressive sensing the matrix $\Phi \in \mathbb{R}^{d\times K}$ is called measurement matrix and models the data acquisition process $y = \Phi x$ with a signal $x \in \mathbb{R}^K$ and the measured data $y \in \mathbb{R}^d$. Rather than to approximate y, the objective is to reconstruct or approximate x, perhaps from a noisy measurement $\tilde{y} = \Phi x + \eta$.
A comprehensive introduction into compressive sensing can be found in [5]. In this thesis, however, we are going to remain mostly in the sparse approximation terminology. After having introduced different algorithms, we are going to look at different types of the key ingredient in sparse approximation: the dictionary.

1.3 Different types of dictionaries

This section discusses different dictionary types, namely random dictionaries, dictionaries consisting of the union of two orthonormal bases, and learned dictionaries.

Random dictionaries

In order to construct a random dictionary matrix Φ, its entries are drawn independently from a fixed distribution, in our case the standard normal distribution, before the columns are normalized. More commonly than in sparse approximation, these matrices are used as measurement matrices in compressive sensing. Simulations and theoretical discoveries, e. g. in [5], have shown that random matrices on average have properties that allow for good performance in compressive sensing, such as a low restricted isometry constant (see Chapter 2).

Dictionaries formed by the union of two orthonormal bases

Since we want to approximate a vector by using only a few atoms, it is reasonable to take the union of two bases as our dictionary, because the vector has an exact representation in each of these bases. Using two orthonormal bases adds, if chosen sensibly, the advantage of low coherence, which can guarantee good performance of sparse approximation algorithms, as we are going to see in Chapter 2.
In the simulations performed in this thesis for example, the Dirac-DCT dictionary is used. This dictionary involves the inverse DCT matrix:

Definition 1.8 (Discrete Cosine Transform matrix)**.**
The *normalized Discrete Cosine Transform matrix of type 2* (DCT) $C \in \mathbb{R}^{d\times d}$ is defined by

$$C_{kj} := \begin{cases} \sqrt{\frac{2}{d}} \cos\left(\left(j-\frac{1}{2}\right)(k-1)\frac{\pi}{d}\right) & \text{for } 1 \le j \le d,\ 2 \le k \le d \\ \frac{1}{\sqrt{d}} & \text{for } k = 1 \end{cases}.$$

C is orthogonal, therefore its inverse is given by $C^{-1} = C^\top$.

The DCT matrix is widely used in the fields of image processing, for example in the jpeg-compression standard (see [9]).

Definition 1.9 (Dirac-DCT dictionary)**.**
The *Dirac-DCT dictionary* is formed by the union of the standard basis of $\mathbb{R}^d$ and the normalized columns of the inverse Discrete Cosine Transform matrix of type 2 (see also [12]).

Learned dictionaries

Since classes of signals (e.g. drum sounds or images of trees) often have structural similarities, it has proven useful to modify a raw dictionary to make it more suitable for a certain signal class in a process called *dictionary learning*. One of the first widely recognized research projects in this field was done by the neuroscientists Olshausen and Field in 1996 (see [7]). Since then, both the statistical learning and the signal processing community have made theoretical and practical advances in dictionary learning. This progress has also been promoted by the greater computational capacities that have been developed in recent years.

There are many different ways of performing dictionary learning, one of these is used by the class of alternating optimization algorithms, which have the following rough structure: The algorithms require the input of a set of training signals, an initialization dictionary, and a sparsity level. In the first step, the sparse approximation of the training signals with the initialization dictionary is calculated using any sparse approximation algorithm. After that the atoms are modified to minimize the approximation error. These two steps can also be repeated a given number of iterations.

One representative of this class of dictionary learning algorithms is the *iterative thresholding and K residual means* algorithm (see [10]), which is used in Chapter 4 of this thesis.

In this first introductory chapter we have defined sparse approximation and its basic concepts. Additionally, different sparse approximation algorithms and different types of dictionaries have been introduced. With this foundation, we can build the theoretical framework further in the next chapter: We are going to formulate dictionary properties that guarantee success in sparse approximation. With these properties, we are going to compare the theoretical worst case performance of the HTP algorithm with other algorithms.

2 Theoretical properties

The second chapter presents theoretical properties of dictionaries that can be used to formulate sufficient conditions for success in sparse approximation or compressive sensing.

2.1 Properties of dictionaries

First, we consider the coherence of a dictionary.

Definition 2.1 (Coherence)**.**
Let $\Phi \in \mathbb{R}^{d\times K}$ be a dictionary matrix. For $j \in \mathbb{N},\ j \leq K$, we are going to denote the jth column of Φ by Φ_j. The *coherence* μ of Φ is defined as

$$\mu := \max_{j\neq i} |\langle \Phi_j, \Phi_i \rangle|.$$

The coherence measures the similarity between atoms in a dictionary. As the columns of a dictionary have unit norm, it follows that $0 \leq \mu \leq 1$.
Coherence conditions in form of upper bounds that guarantee success in sparse approximation have been developed for several sparse approximation algorithms. If $\mu < \frac{1}{2s-1}$, for example, both OMP and BP are guaranteed to recover the exact sparse representation of an s-sparse signal in the dictionary (see [13]).
However, the coherence of a dictionary cannot be arbitrarily small, as we are going to see in the following proposition.

Proposition 2.2 (Welch bound)**.**
Let $\Phi \in \mathbb{R}^{d\times K}$ be a dictionary matrix. The coherence μ of Φ satisfies the lower bound

$$\mu \geq \sqrt{\frac{K-d}{d(K-1)}}.$$

Proof. This proof follows the steps of the proof of a more general statement in [15].
Define the auxiliary term

$$B := \sum_{i,j=1}^{K} |\langle \Phi_i, \Phi_j \rangle|^2.$$

We are going to find an upper and a lower bound on B, which is going to help us to bound the coherence of Φ.

An upper bound on B can be constructed easily:

$$B = \sum_{i,j=1}^{K} |\langle \Phi_i, \Phi_j \rangle|^2 = \sum_{i=1}^{K} |\langle \Phi_i, \Phi_j \rangle|^2 + \sum_{\substack{i,j=1 \\ i \neq j}}^{K} |\langle \Phi_i, \Phi_j \rangle|^2 \leq$$
$$\leq \sum_{i=1}^{K} 1^2 + \sum_{\substack{i,j=1 \\ i \neq j}}^{K} \mu^2 = K + K(K-1)\mu^2$$

For a lower bound on B we have to work harder. In a first step we rearrange the sum to obtain

$$\begin{aligned} B = \sum_{i,j=1}^{K} |\langle \Phi_i, \Phi_j \rangle|^2 &= \sum_{i,j=1}^{K} \left(\sum_{l=1}^{d} \Phi_{li} \Phi_{lj} \right) \left(\sum_{n=1}^{d} \Phi_{ni} \Phi_{nj} \right) = \\ &= \sum_{i,j=1}^{K} \sum_{l,n=1}^{d} \Phi_{li} \Phi_{lj} \Phi_{ni} \Phi_{nj} = \\ &= \sum_{l,n=1}^{d} \sum_{i,j=1}^{K} \Phi_{li} \Phi_{lj} \Phi_{ni} \Phi_{nj} = \\ &= \sum_{l,n=1}^{d} \left(\sum_{i=1}^{K} \Phi_{il} \Phi_{in} \right)^2 = \\ &= \sum_{l=1}^{d} \left(\sum_{i=1}^{K} \Phi_{il} \Phi_{il} \right)^2 + \sum_{\substack{l,n=1 \\ l \neq n}}^{d} \left(\sum_{i=1}^{K} \Phi_{il} \Phi_{in} \right)^2 \geq \sum_{l=1}^{d} \left(\sum_{i=1}^{K} \Phi_{il}^2 \right)^2 . \end{aligned}$$

Now we employ the Cauchy-Schwarz inequality to obtain

$$B \geq \frac{\left(\sum_{l=1}^{d} \sum_{i=1}^{K} \Phi_{il}^2 \cdot 1 \right)^2}{\sum_{l=1}^{d} 1} = \frac{\left(\sum_{i=1}^{K} \sum_{l=1}^{d} \Phi_{il}^2 \right)^2}{d} = \frac{\left(\sum_{j=1}^{K} 1 \right)^2}{d} = \frac{K^2}{d}.$$

Finally we bring the upper and the lower bound on B together:

$$K + K(K-1)\mu^2 \geq B \geq \frac{K^2}{d},$$

which leads us to

$$\mu^2 \geq \frac{K-d}{d(K-1)},$$

from which we derive the statement of the proposition. □

An attempt to construct incoherent dictionaries is to form the union of two or more orthonormal bases, because the coherence of each of the bases is zero. Therefore the coherence of such a dictionary is determined only by the inner products between atoms of different bases. Consider for example the Dirac-DCT dictionary:

Example 2.3 (Coherence of the Dirac-DCT dictionary).
The coherence of the Dirac-DCT dictionary matrix can be calculated as follows: Both of the bases are orthogonal, so the coherence μ is the largest (in modulus) entry of the normalized DCT matrix C.

$$\mu = \max_{j \neq i} |\langle \Phi_j, \Phi_i \rangle| = \max_{j,i} |\langle e_i, C_{-j} \rangle| = \max_{j,i} |C_{ij}| =$$
$$= \max_{\substack{1 \leq j \leq d \\ 2 \leq i \leq d}} \left\{ \left| \sqrt{\frac{2}{d}} \cos\left(\left(j - \frac{1}{2} \right) (i-1) \frac{\pi}{d} \right) \right|, \frac{1}{\sqrt{d}} \right\}$$

From this we derive

$$\frac{1}{\sqrt{d}} \leq \mu \leq \sqrt{\frac{2}{d}}.$$

The Dirac-DCT dictionary achieves the upper bound in many dimensions d, e. g. in $d = 128$, which is the dimension of the simulated signals in Chapter 3 of this thesis.

Another dictionary (or measurement matrix) property is used in compressive sensing, namely the restricted isometry property described by the restricted isometry constant.

Definition 2.4 (Restricted isometry constant (RIC))**.**
Let $\Phi \in \mathbb{R}^{d \times K}$ be a dictionary matrix, and $s << K$.
The *s-th order restricted isometry constant* $\delta_s = \delta_s(\Phi)$ of Φ is defined as

$$\delta_s := \inf\{\delta \geq 0 \colon (1-\delta)\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1+\delta)\|x\|_2^2 \text{ for all } s\text{-sparse } x \in \mathbb{R}^K\}.$$

This property characterizes the conditioning of column submatrices of the dictionary matrix. Similarly to the role of coherence in sparse approximation, RIC conditions in form of upper bounds have been formulated that guarantee success in compressive sensing settings for several algorithms (e.g. in [5]).
The next lemma connects the concepts of coherence and restricted isometry. In a slightly different version it can be found in [5] (Theorem 5.3).

Lemma 2.5 (Relation between coherence and restricted isometry)**.**
Let $\Phi \in \mathbb{R}^{d\times K}$ be a dictionary matrix with normalized columns and $s << K$. Then the coherence μ and the restricted isometry constant δ_s of Φ satisfy

$$\delta_s \leq (s-1)\mu.$$

Proof. We have to show that for all s-sparse $x \in \mathbb{R}^K$

$$(1-(s-1)\mu)\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1+(s-1)\mu)\|x\|_2^2. \tag{2}$$

Let $x \in \mathbb{R}^K$ be s-sparse with index set I. Let Φ_I be the matrix consisting of the columns of Φ with indices in I, similarly $x_I \in \mathbb{R}^s$ contains only the entries of x with indices in I, i.e. the non-zero entries of x.
We obtain for the middle part of the inequality

$$\|\Phi x\|_2^2 = \langle \Phi x, \Phi x\rangle = \langle \Phi_I x_I, \Phi_I x_I\rangle = \langle x_I, \Phi_I^\top \Phi_I x_I\rangle. \tag{3}$$

The matrix $\Phi_I^\top \Phi_I$ is real and symmetrical and therefore, according to the spectral theorem, there is an orthonormal basis $(v_\lambda)_{\lambda\in\Lambda}$ of $\mathbb{R}^s$ consisting of eigenvectors of $\Phi_I^\top\Phi_I$ and all the eigenvalues $\lambda \in \Lambda$ of $\Phi_I^\top\Phi_I$ are real. Additionally, the eigenvalues are nonnegative because $\Phi_I^\top\Phi_I$ is positive semidefinite.
If we write x_I as a linear combination of the eigenbasis $x_I = \sum_{\lambda\in\Lambda} c_\lambda v_\lambda$, we can continue with (3):

$$\langle x_I, \Phi_I^\top\Phi_I x_I\rangle = \sum_{\lambda\in\Lambda} c_\lambda \langle x_I, \Phi_I^\top \Phi_I v_\lambda\rangle = \sum_{\lambda\in\Lambda} c_\lambda \langle x_I, \lambda v_\lambda\rangle$$

With $\lambda_{\max}$ and $\lambda_{\min}$ denoting the largest and smallest eigenvalues of $\Phi_I^\top\Phi_I$, we obtain:

$$\lambda_{\min}\|x\|_2^2 = \lambda_{\min}\sum_{\lambda\in\Lambda} c_\lambda\langle x_I, v_\lambda\rangle \leq \sum_{\lambda\in\Lambda} c_\lambda\langle x_I, \lambda v_\lambda\rangle \leq \lambda_{\max}\sum_{\lambda\in\Lambda} c_\lambda\langle x_I, v_\lambda\rangle = \lambda_{\max}\|x\|_2^2,$$

and therefore

$$\lambda_{\min}\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq \lambda_{\max}\|x\|_2^2.$$

In order to prove the inequalities (2), we have to show that

$$\forall\ \lambda \in \Lambda : 1-(s-1)\mu \leq \lambda \leq 1+(s-1)\mu. \tag{4}$$

This can be accomplished with the help of the Gershgorin circle theorem: In application to our situation it states that

$$\forall\ \lambda \in \Lambda : \exists\ i \in I : \langle \Phi_i, \Phi_i \rangle - \sum_{\substack{j \in I \\ i \neq j}} |\langle \Phi_i, \Phi_j \rangle| \leq \lambda \leq \langle \Phi_i, \Phi_i \rangle + \sum_{\substack{j \in I \\ i \neq j}} |\langle \Phi_i, \Phi_j \rangle|$$

because the (i,j)th entry of $\Phi_I^\top \Phi_I$ is $\langle \Phi_i, \Phi_j \rangle$.
By definition of μ we can calculate

$$\sum_{\substack{j \in I \\ i \neq j}} |\langle \Phi_i, \Phi_j \rangle| \leq \sum_{\substack{j \in I \\ i \neq j}} \mu = (s-1)\mu.$$

With this, we have shown inequality (4). (Note that the columns of Φ are normalized, i.e. $\langle \Phi_i, \Phi_i \rangle = 1$.) □

With this connection between coherence and restricted isometry, restricted isometry conditions taken from the compressive sensing literature can be translated into coherence conditions in order to suit the theoretical framework of sparse approximation. In the following chapter for example, an error estimation theorem for HTP from [4] involving a restricted isometry condition is reformulated into a statement about incoherent dictionaries.

2.2 Sparse approximation with an incoherent dictionary matrix

The following proposition is a modified version of Theorem 3.8 in [4].

Proposition 2.6 (Sparse approximation with HTP)**.**
Let $\Phi \in \mathbb{R}^{d \times K}$ be a dictionary matrix with normalized columns and coherence

$$\mu < \frac{1}{\sqrt{3}(3s-1)} \tag{5}$$

for a given sparsity level $s \ll K$.
Then for any s-sparse $x \in \mathbb{R}^K$ and any $\eta \in \mathbb{R}^d$, the sequence $(x^n)_{n \in \mathbb{N}}$ defined by the iterations of the HTP algorithm with $y = \Phi x + \eta$ satisfies

$$\|x^n - x\|_2 \leq \rho^n \|x^0 - x\|_2 + \tau \frac{1-\rho^n}{1-\rho} \|\eta\|_2$$

with constants $\rho < 1$ and $\tau \leq 5.15$.

Proof. The statement follows directly by applying Theorem 3.8 in [4] to s-sparse x. Note that $\delta_{3s} < \frac{1}{\sqrt{3}}$ follows from $\mu < \frac{1}{\sqrt{3}(3s-1)}$ according to Lemma 2.5. □

The result presented in Proposition 2.6 is rather disappointing because in order to satisfy the bound (5), s has to be impractically small. However, other sparse approximation algorithms have similar theoretical limitations. In [13] for example, Tropp develops several statements about the performance of the OMP algorithm for sparse approximation with upper bounds on the coherence from $\frac{1}{2s}$ to $\frac{1}{3s}$. Although these are more generous than the bound in Proposition 2.6, they are still hardly achievable in practice.

A different way to evaluate the performance of an algorithm is to look at its average performance (as opposed to its worst case performance considered earlier). This can be done either by simulations or by theoretical analysis. The latter has been done successfully for thresholding (in [11]) and BP (in [14]). Although OMP performs well in simulation studies, there is no theoretical proof for its average performance yet.

Because of its similarity to thresholding it is reasonable to see HTP as a good candidate for a theoretical average case performance. As a start in this direction, this thesis presents average performance simulation results of HTP for sparse recovery and sparse approximation in the following chapter.

3 Simulations

In the following section we are going to look at the results of simulations performed with the HTP and the OMP algorithm.
We present two types of simulations: exact sparse recovery (with constant and non-constant coefficients) and sparse approximation. These problems are defined in Chapter 1.1. Let us consider more specifically what exact sparse recovery and sparse approximation mean in the following simulations:

Exact sparse recovery
In this setting the signal y is s-sparse in the dictionary Φ, i.e. it can be written as a linear combination of s atoms:

$$y = \Phi x, \text{ with } \|x\|_0 \leq s.$$

The goal of exact sparse recovery is to recover the coefficients of y in Φ, i.e. to find x.

Sparse approximation
Let $\eta \in \mathbb{R}^d$ denote the noise or noise vector. In sparse approximation, η is added to an s-sparse signal:

$$\tilde{y} = \Phi x + \eta, \text{ with } \|x\|_0 \leq s \text{ and } \|\eta\|_2 \text{ small.}$$

In general an s-sparse representation of $\tilde{y}$ in Φ does not necessarily exist. Instead, the goal is to find a good s-sparse approximation of $\tilde{y}$.

The simulations in both settings are conducted as follows:
They are done with two different dictionaries of size 128×256, namely a random dictionary and the Dirac-DCT dictionary.
For s varying from 1 to 64, 500 s-sparse signals are generated in the dictionary according to a methodology specific to the setting explained below. The s-sparse approximations of these signals, as calculated by both algorithms, are compared on the basis of different performance measures (e.g. error norm, success rate).

3.1 Exact sparse recovery with constant coefficients

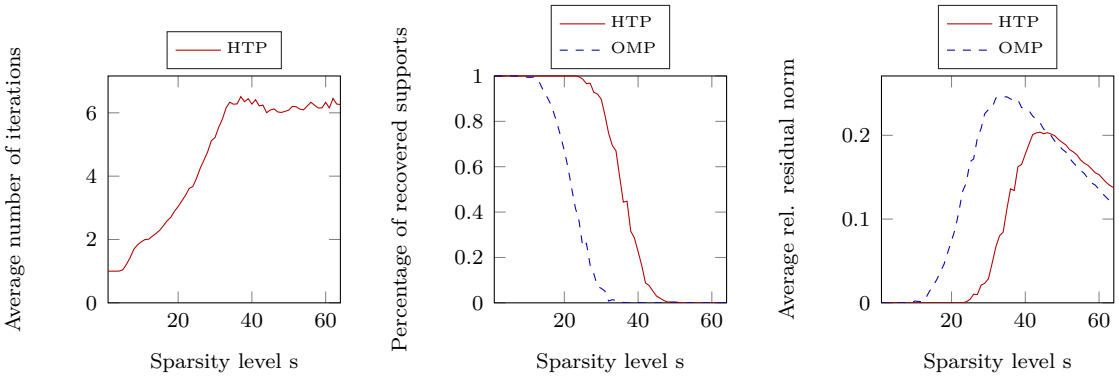
In the exact-sparse simulations with constant coefficients the s-sparse signals y are generated in the dictionary Φ as follows: The elements of the support set I of $x \in \mathbb{R}^K$ are drawn randomly from $\{1, \ldots, K\}$ and the signal is constructed according to:

$$y := \Phi x, \text{ with } x_i \sim \mathcal{U}_{\{-\frac{1}{\sqrt{s}}, \frac{1}{\sqrt{s}}\}} \text{ i.i.d. if } i \in I, \text{ and } x_i := 0 \text{ otherwise.}$$

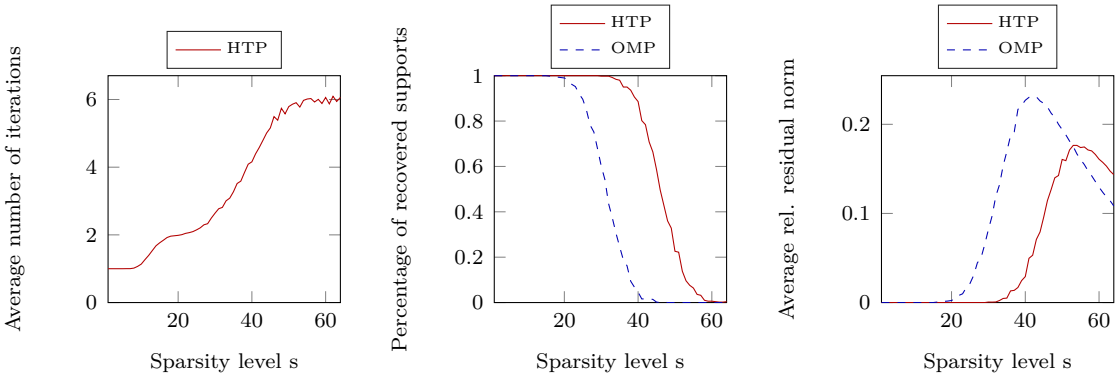
(The abbreviation *i.i.d.* stands for *independent and identically distributed.*)
Note that by this construction we ensure that $\|x\|_2 = 1$.
As measures of performance for each sparsity level we use the percentage of correctly recovered supports, the average relative residual norm, and the average number of iterations. The number of iterations is a meaningful measure only for HTP because the OMP algorithm always goes through s iterations.

Results

The results of the exact-sparse recovery simulations are shown in Figure 1. The overall impression is that HTP performs better in this setting than OMP: Both the success rates and the residual error lead to this conclusion for both dictionaries.
When taking a closer look at the results, it is clear that both algorithms perform better with the Dirac-DCT dictionary than with a random dictionary matrix. We are going to see this difference in all the following simulation settings too.
The average number of iterations completed by the HTP algorithm is rather low (mostly under 6 iterations) in both settings. This is significantly lower than the s iterations the OMP algorithm goes through. Although one iteration of HTP takes longer than one of OMP, the smaller number of iterations leads to a shorter average time per reconstruction of HTP compared to the time needed by OMP, especially when s is not too small (see Figure 2).
In these simulations we have operated with signals that have an exact s-sparse representation in the dictionary with constant coefficients. In this setting, the HTP algorithm clearly outperforms the OMP algorithm. In the following simulation, we are going to alter the setting to work with signals whose s-sparse representation has non-constant coefficients.

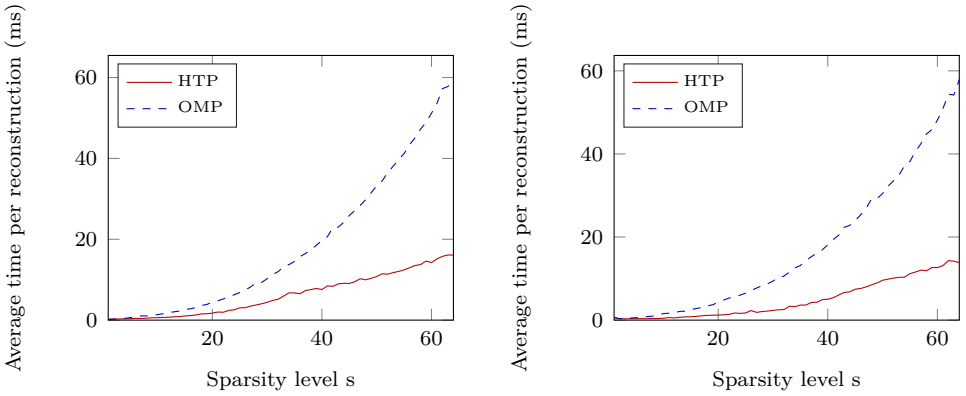


(a) Random dictionary matrix



(b) Dirac-DCT dictionary

Figure 1: Simulation results: Exact-sparse recovery with constant coefficients ($d = 128,\ K = 256$)



(a) Random dictionary (b) Dirac-DCT dictionary

Figure 2: Runtime analysis: Exact-sparse recovery with constant coefficients ($d = 128,\ K = 256$)

3.2 Exact sparse recovery with non-constant coefficients

Until now we have looked at sparse signals that were constructed with s constant (in modulus) coefficients. In a next step we alter this methodology to achieve a decay in the coefficients.

Let $0 < b < 1$ be a decay parameter. The indices $j_1, \ldots, j_s$ in the support set I of $x \in \mathbb{R}^K$ are drawn randomly from $\{1, \ldots, K\}$. The coefficients with indices not in I are set to zero, i. e.

$$x_i := 0 \text{ for } i \notin I.$$

For the other coefficients we follow a two step construction. First $\tilde{x}_i$ is drawn according to

$$\tilde{x}_{j_k} \sim \mathcal{U}_{\{-\frac{1}{\sqrt{s}}, \frac{1}{\sqrt{s}}\}} \text{ for } 1 \leq k \leq s.$$

Then the coefficients are resized according to the decay parameter as follows:

$$x_{j_k} := c(1-b)^k \tilde{x}_{j_k} \text{ for } 1 \leq k \leq s$$
$$\text{with } c = \frac{\sqrt{s}}{1-b} \sqrt{\frac{1-(1-b)^2}{1-(1-b)^{2s}}} \text{ for normalization}$$

Finally the signal y is constructed by

$$y := \Phi x.$$

This construction ensures that we have s nonzero coefficients with exponential decay and uniformly distributed signs.

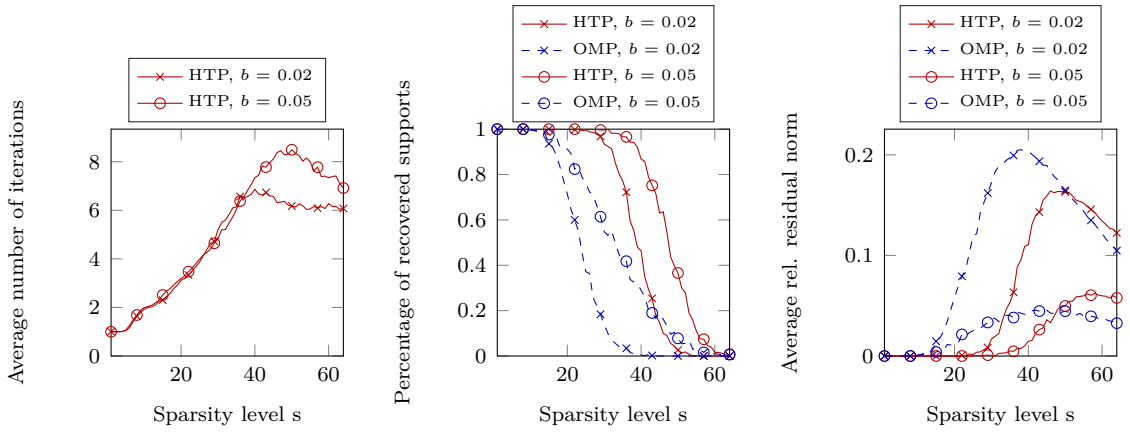
Results

The results of this simulation are shown in Figure 3. The introduction of decay improves the performance of both algorithms as measured by success rate and residual norm.

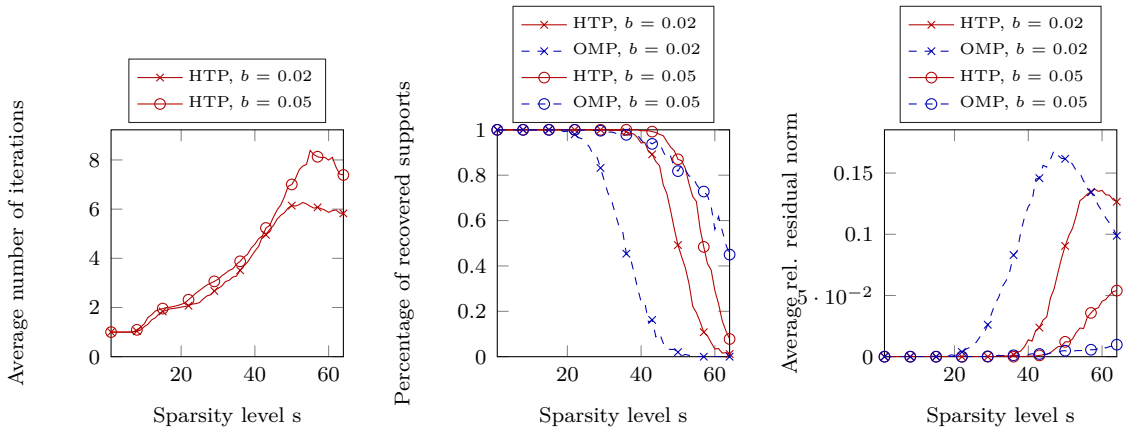
In the random dictionary setting the HTP algorithm shows better performance than the OMP algorithm. This can also be observed in the Dirac-DCT dictionary setting for moderate decay. Only when confronted with strong decay OMP outperforms HTP in the Dirac-DCT dictionary setting. This allows the interpretation that due to its greedy nature, OMP can pick up better on stronger contributions of individual atoms to the signal compared to a signal with constant coefficients. The HTP algorithm on the other

hand does not build the support incrementally one atom after another but picks a whole package at once. Therefore it has a harder time identifying smaller factors in the signal. The average number of iterations of HTP is similar to those in the constant-coefficient setting. However, when the sparsity level and the decay are high, the average number of iterations of the HTP algorithm reaches its peak at 8 iterations. Although this is higher than the number of iterations needed when confronted with constant coefficients or moderate decay, it is still considerably less than the number of iterations needed by OMP.

So far, the simulations have only included signals that have exact s-sparse representations in a given dictionary. We have seen that the HTP algorithm can more than keep up with the OMP algorithm in most of these exact s-sparse recovery settings. The following simulations are going to look at signals that do not have an exact s-sparse representation in the dictionary, which will be more of a challenge to both algorithms.



(a) Random dictionary matrix



(b) Dirac-DCT dictionary

Figure 3: Comparing HTP and OMP: Exact-sparse recovery with decreasing coefficients (decay parameter b, $d = 128,\ K = 256$)

3.3 Sparse approximation

The exact-sparse setting as explained above, is an academic one: In real world applications one will rarely be confronted with signals that have an exact s-sparse representation in the given dictionary. More often those signals will not be exactly sparse by nature or distorted by some kind of noise, which we are going to simulate in the following section.
The raw s-sparse signal y is generated as we have seen in the exact sparse setting with constant coefficients.

$$y := \Phi x, \text{ with } x_i \sim \mathcal{U}_{\{-\frac{1}{\sqrt{s}}, \frac{1}{\sqrt{s}}\}} \text{ i.i.d. if } i \in I \text{ and } x_i := 0 \text{ otherwise}$$

For the sparse approximation simulations a randomly generated noise vector η of noise level ρ is added to y in order to construct the noisy signal $\tilde{y}$ for the algorithm's input:

$$\tilde{y} := y + \eta, \text{ with } \eta_i \sim \mathcal{N}(0, \frac{\rho^2}{d}) \text{ i.i.d.}$$

As we can see in the construction above, the noise level ρ influences how much noise is added to the signal. However, in order to quantify the relation between signal and noise, we define the energy of a signal E and the signal-to-noise ratio SNR:

Definition 3.1 (Energy of a signal)**.**
Let $y \in \mathbb{R}^d$ be a signal. The *energy* of y is given by

$$E_y := \|y\|_2^2.$$

Definition 3.2 (Signal-to-noise ratio)**.**
Let $y \in \mathbb{R}^d$ be a signal and consider $\eta \in \mathbb{R}^d$ as noise. If generated by a random process we will consider y and η as random variables.
The *signal-to-noise ratio* SNR of the noisy signal $\tilde{y} = y + \eta$ is defined as

$$SNR := \frac{\mathbb{E}(E_y)}{\mathbb{E}(E_\eta)}.$$

In order to calculate the energy of the signal $y = \Phi x$ we are going to consider a more general setting for the signal construction under the assumptions:

- $x_1, \ldots, x_K$ independently distributed,
- $\mathbb{E}(\|x\|_2^2) = 1$ and $\forall\ i = 1, \ldots, K \colon \mathbb{E}(x_i) = 0$
- $\forall\ j = 1 \ldots d \colon \|\Phi_j\|_2^2 = 1$

Note that the last assumption is true by definition for all dictionaries.
Under these conditions we derive the following result for the expected value of the signal energy:

$$\begin{aligned}
\mathbb{E}(E_y) = \mathbb{E}\|y\|_2^2 = \mathbb{E}\left(\sum_{i=1}^{d} y_i^2\right) = \sum_{i=1}^{d} \mathbb{E}(y_i^2) = \sum_{i=1}^{d} \mathbb{E}\left(\sum_{j=1}^{K} \Phi_{ij} x_j\right)^2 = \\
= \sum_{i=1}^{d} \mathbb{E}\left(\sum_{j=1}^{K} \Phi_{ij}^2 x_j^2 + \sum_{\substack{k,l=1 \\ k\neq l}}^{K} \Phi_{ik}\Phi_{il} x_k x_l\right) = \\
= \sum_{i=1}^{d} \left(\sum_{j=1}^{K} \Phi_{ij}^2 \mathbb{E}(x_j^2) + \sum_{\substack{k,l=1 \\ k\neq l}}^{K} \Phi_{ik}\Phi_{il} \mathbb{E}(x_k x_l)\right) = \\
= \sum_{i=1}^{d}\sum_{j=1}^{K} \Phi_{ij}^2 \mathbb{E}(x_j^2) + \sum_{i=1}^{d}\sum_{\substack{k,l=1 \\ k\neq l}}^{K} \Phi_{ik}\Phi_{il} \mathbb{E}(x_k)\mathbb{E}(x_l) = \\
= \sum_{j=1}^{K} \mathbb{E}(x_j^2) \sum_{i=1}^{d} \Phi_{ij}^2 + 0 = \sum_{j=1}^{K} \mathbb{E}(x_j^2)\|\Phi_j\|_2^2 = \sum_{j=1}^{K} \mathbb{E}(x_j^2) = \mathbb{E}(\|x\|_2^2) = 1
\end{aligned}$$

The energy of the noise follows from its distribution:

$$\mathbb{E}(E_\eta) = \mathbb{E}\|\eta\|_2^2 = \mathbb{E}\left(\sum_{i=1}^{d} \eta_i^2\right) = \sum_{i=1}^{d} \mathbb{E}(\eta_i^2) = \sum_{i=1}^{d} \frac{\rho^2}{d} = \rho^2$$

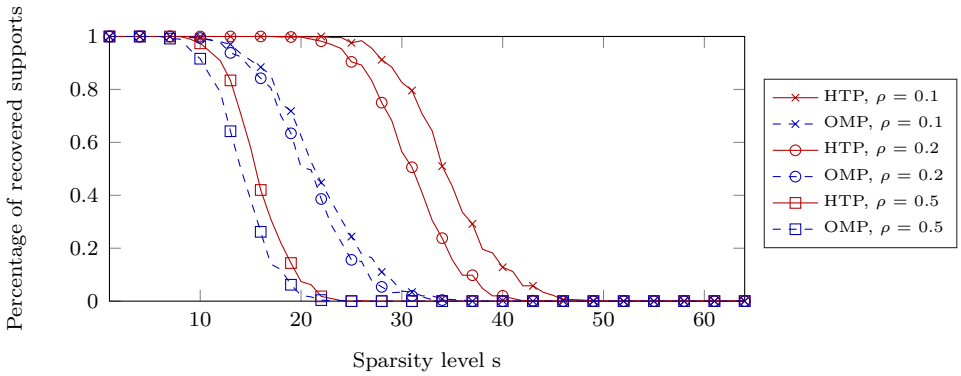
Now we obtain the signal-to-noise ratio

$$SNR = \frac{\mathbb{E}(E_y)}{\mathbb{E}(E_\eta)} = \frac{1}{\rho^2}.$$

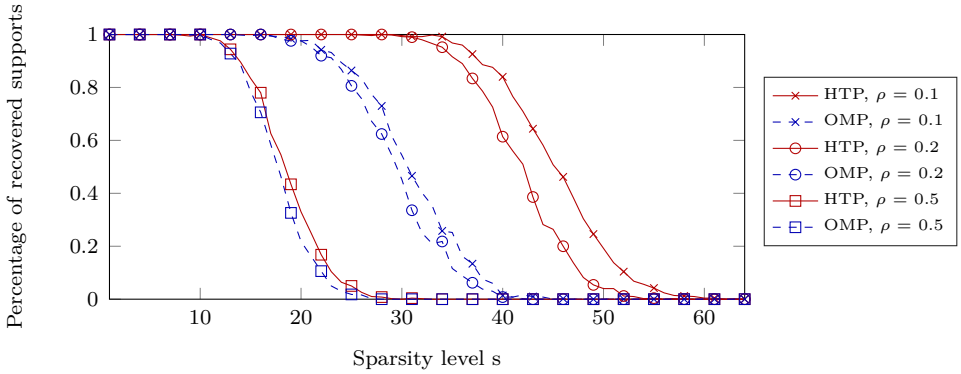
In the simulations we are going to compare three different noise levels which are given in Table 1.

ρ	SNR
0.1	100
0.2	25
0.5	4

Table 1: Noise levels and their corresponding SNRs



(a) Random dictionary matrix



(b) Dirac-DCT dictionary

Figure 4: Simulation results: Sparse approximation with constant coefficients (noise level ρ, $d = 128$, $K = 256$)

Results

The results of the sparse approximation simulations are shown in Figure 4.

Similar to what we have seen in the exact-sparse recovery setting with constant coefficients, the HTP algorithm gives better results than the OMP algorithm for sparse approximation. Only at a noise level of $\rho = 0.5$ ($SNR = 4$) the difference between the two algorithms vanishes and they perform equally poorly corresponding to the very low SNR.

Overall, both algorithms cope well with moderate noise levels.

In this chapter, we have observed the performance of the OMP and HTP algorithms in different simulation settings: exact s-sparse recovery with constant and non-constant coefficients and sparse approximation. In most if these settings, the success rate of HTP exceeds the success rate of OMP. Furthermore, on average, the HTP algorithm needs only a few iterations to construct such a approximation. In the following chapter, we are abandoning the simulation setting with its synthetically constructed signals and working with real-life picture data instead.

4 An image processing example

In the last chapter of this thesis, both the HTP and the OMP algorithm are tested with real life data instead of the synthetic signals we have seen in the simulations in Chapter 3.
For this purpose a grayscale image of pixel size 245×245 (see Figure 5) is deconstructed into overlapping image patches of pixel size 8×8. These signals in $\mathbb{R}^{8\times 8}$ are sparsely approximated with different dictionaries following the methodology explained below.
The performance of the two algorithms in this approximation task is measured by taking the average error norm of the resulting approximated image patches as compared to the original ones.

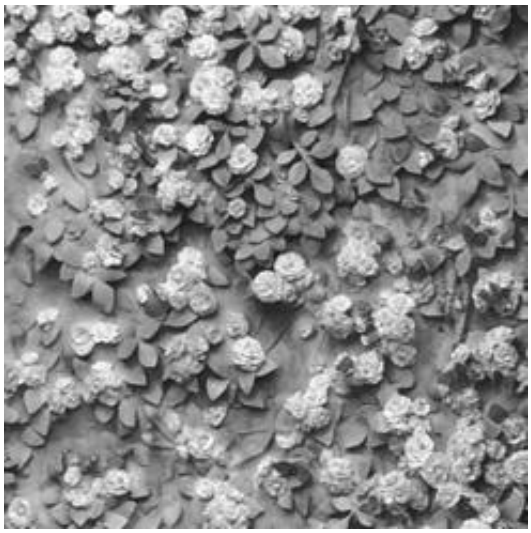


Figure 5: Image used in Chapter 4

4.1 Two-dimensional sparse approximation

So far we have looked at the signal space $\mathbb{R}^d$, which we are going to call the *one-dimensional* signal space by abuse of terminology, in contrast to the *two-dimensional* signal space $\mathbb{R}^{d\times d}$. In order to be able to operate with two-dimensional signals we first have to define the Kronecker product basis.

Definition 4.1 (Kronecker product)**.**
Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{k\times l}$ be matrices. The *Kronecker product* $A \otimes B \in \mathbb{R}^{mk\times nl}$ of A and B is defined as

$$A \otimes B := \begin{bmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{bmatrix}.$$

Definition 4.2 (Kronecker product basis)**.**
Let $\underline{v} = (v_j)_{1\leq j\leq d}$ and $\underline{w} = (w_k)_{1\leq k\leq d}$ be two bases of $\mathbb{R}^d$. The elements of the *Kronecker product basis* $\underline{v} \otimes \underline{w} := (u_l)_{1\leq l\leq d^2}$ of $\underline{v}$ and $\underline{w}$ are defined as

$$u_{j+(k-1)d} := v_j \otimes w_k;\ 1 \leq j, k \leq d.$$

Following this definition, $\underline{v} \otimes \underline{w}$ is indeed a basis of the vector space $\mathbb{R}^{d\times d}$.
If $\underline{v} = \underline{w}$, we denote $\underline{v}^{\otimes 2} := \underline{v} \otimes \underline{v}$.

In analogy to their one-dimensional counterparts we can construct two-dimensional dictionaries in $\mathbb{R}^{d\times d}$.

Definition 4.3 (Two-dimensional dictionaries)**.**
Let d and K be positive integers with $d^2 \leq K$. A family $\mathcal{D} = (\Phi_j)_{1\leq j\leq K}$ of matrices in $\mathbb{R}^{d\times d}$ is called a *two-dimensional dictionary* if its elements satisfy

$$\forall\ 1 \leq j \leq K\colon \|\Phi_j\|_F = 1,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This particular norm is used because it is compatible with the Euclidean norm of vectorized atoms.
Typical examples are random dictionaries and combinations of Kronecker product bases.

Definition 4.4 (Vectorization)**.**
Let $v \in \mathbb{R}^{d\times d}$ be a two-dimensional signal. The *vectorized* version $\tilde{v} \in \mathbb{R}^{d^2}$ of v is defined by

$$\tilde{v}_{j+(k-1)d} := v_{jk},\ 1 \leq j, k \leq d.$$

Vectorization grants a natural isomorphic connection between $\mathbb{R}^{d\times d}$ and $\mathbb{R}^{d^2}$.Therefore, we can apply the sparse approximation theory onto a two-dimensional signal space. With this theoretical foundation, we can introduce different bases and dictionaries in the next section, with which we are going to approximate the image patches.

4.2 Bases and dictionaries

In the following sparse approximation task we are going to compare sparse approximation of HTP and OMP with different bases and dictionaries: the Haar basis and the DCT basis, the Haar-DCT dictionary, and a learned dictionary with random initialization.

Haar basis

The one-dimensional Haar basis $\underline{H}$ of $\mathbb{R}^d$ consists of the normalized columns of the inverse Haar matrix.

Definition 4.5 (Haar matrix)**.**
Let d be a power of 2. The *Haar matrix* $H^{(d)} \in \mathbb{R}^{d \times d}$ is defined recursively by

$$H^{(2)} := \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$
$$H^{(2n)} := \begin{bmatrix} H^{(n)} \otimes [1, 1] \\ I \otimes [1, -1] \end{bmatrix} \text{ for } n \in \mathbb{N}$$

with I denoting the $n \times n$ identity matrix and $\otimes$ denoting the Kronecker product. The normalized Haar matrix is orthogonal.

Since the approximation task at hand is two-dimensional, the vectorized version of the Kronecker product basis $\underline{H}^{\otimes 2}$ of the Haar basis is used.

DCT basis

The one-dimensional DCT basis $\underline{C}$ of $\mathbb{R}^d$ consists of the columns of the inverse DCT matrix (see Definition 1.8), which are already normalized.
As with the Haar basis, in the following section the vectorized version of the Kronecker product basis $\underline{C}^{\otimes 2}$ of the DCT basis is used.

Haar-DCT dictionary

The *raw two-dimensional Haar-DCT dictionary* is the union of the above mentioned two-dimensional bases. This dictionary is visualized in Figure 6.
The coherence of the raw Haar-DCT dictionary is $\mu = 1$, because both of the bases contain a constant atom. Apart from the constant atoms there are also other pairs of Haar and DCT atoms that are very similar, leading to a high coherence.
Because of that, the *incoherent two-dimensional Haar-DCT dictionary* is constructed by removing Haar atoms from the raw Haar-DCT dictionary until a given coherence

threshold is met. In this particular case the threshold is set at $\frac{1}{\sqrt{2}}$, which amounts to the removal of 4 out of 128 atoms (in Figure 6 these removed atoms are marked).

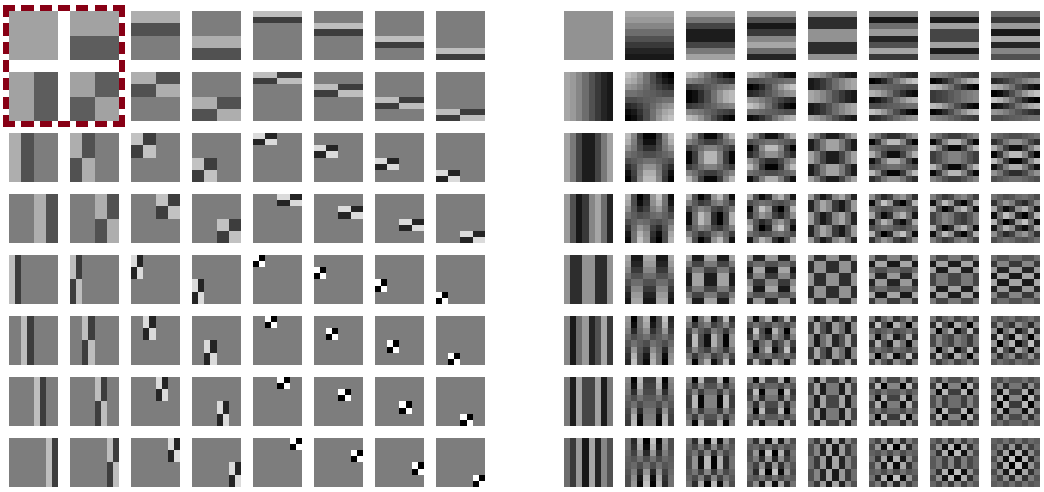
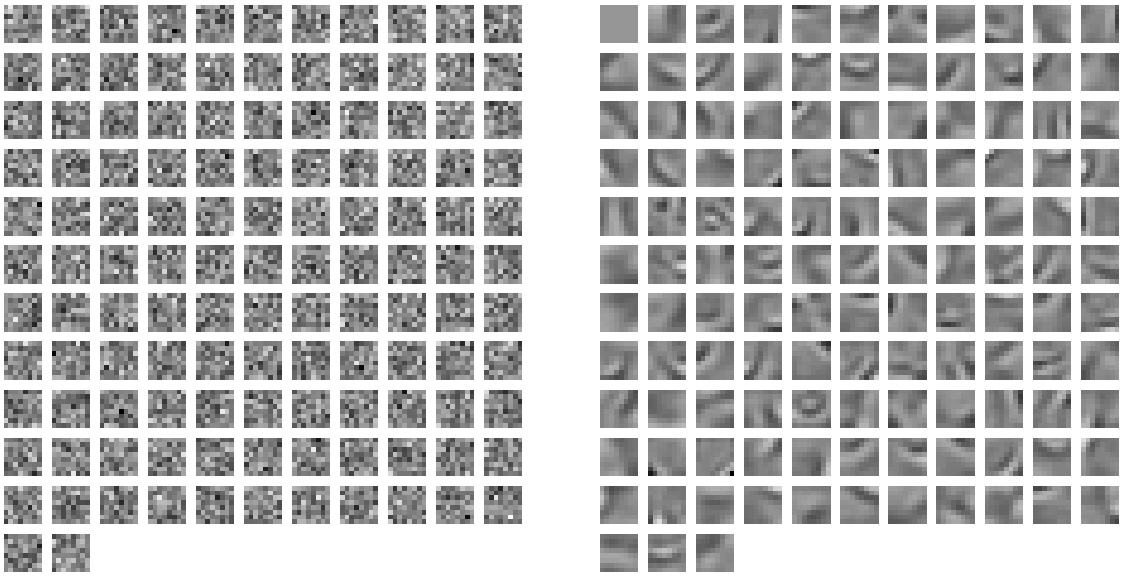


Figure 6: Haar basis (left) and DCT basis (right) forming the Haar-DCT dictionary (marked Haar atoms are removed in the incoherent dictionary)



(a) Random dictionary

(b) Learned dictionary incl. constant atom

Figure 7: Dictionary learning initialization and result (Sparsity level: $5 + 1$)

Learned dictionary

The last dictionary used in this chapter is a learned dictionary with random initialization constructed in the following setting:
Firstly, the patches are normalized before the dictionary learning process.
Secondly, another patch modification is advisable due to a property of the image patches used: They have a strong contribution of a constant factor, i. e. the inner product between each patch and a constant atom is high. If we used the unaltered patches to train the dictionary, this strong constant factor would overshadow other weaker factors. Since we already know that a constant atom will be required in the dictionary, the constant factor is removed from the patches for the duration of the learning process and a constant atom is added to the learned dictionary in the end. Similar measures have been taken for the same purpose in the natural image processing literature (e.g. in [1]).
After these preparations, the dictionary learning process is initialized using a random dictionary of 123 atoms $\in \mathbb{R}^{64}$, whose entries are drawn from a standard normal distribution before normalization, see Figure 7a. This initialization dictionary is modified using the iterative thresholding and K residual means method (ITKrM) described in [10]. The algorithm is iterated 50 times and each time takes a random subset of 10000 of the total 56644 patches to train the dictionary.
This dictionary learning scheme is carried out with two different sparsity levels, 5 and 7 (which amounts to $5+1=6$ and $7+1=8$ counting the constant atom). The resulting learned dictionary with sparsity level $5+1$ is visualized in Figure 7b.
Now, we are going to sparsely approximate the image patches with the bases and dictionaries described above and observe if the HTP algorithm performs well in this image data setting too.

4.3 Results

The results are shown in Figure 8.
The poorest approximation is done with the Haar basis. This can be explained by the smooth structure of the image patches in contrast to the spiky Haar atoms. With the more compatible properties of the DCT basis the patches can be approximated in a better way, nearly as well as with the Haar-DCT dictionary. The difference between those two is disappointingly small and does not justify the greater effort needed for sparse approximation with a dictionary.
However, we see an improvement with the learned dictionary of sparsity level $5+1$. This observation is not surprising because this dictionary has been modified to fit the purpose. The advantage of the learned dictionary is largest when the sparsity level for

which it has been trained is reached and decreases again with increasing sparsity levels. OMP performs slightly better than HTP in the learned dictionary setting, which can be explained by the non-constant coefficients of the patches in the dictionary (similar to the simulation setting in Chapter 3.2). On the other hand, the average time per patch reconstruction of the HTP algorithm is considerably lower than the time needed by the OMP algorithm, which is shown in the runtime analysis (see Figure 9).
Note that in Figure 8 the results of the learned dictionary setting with sparsity level $7 + 1$ are not shown because the results are very similar to those of the other learned dictionary setting.
This image processing example shows that apart from algorithm selection for sparse approximation, choosing (or constructing) a suitable dictionary for the task at hand is very important.

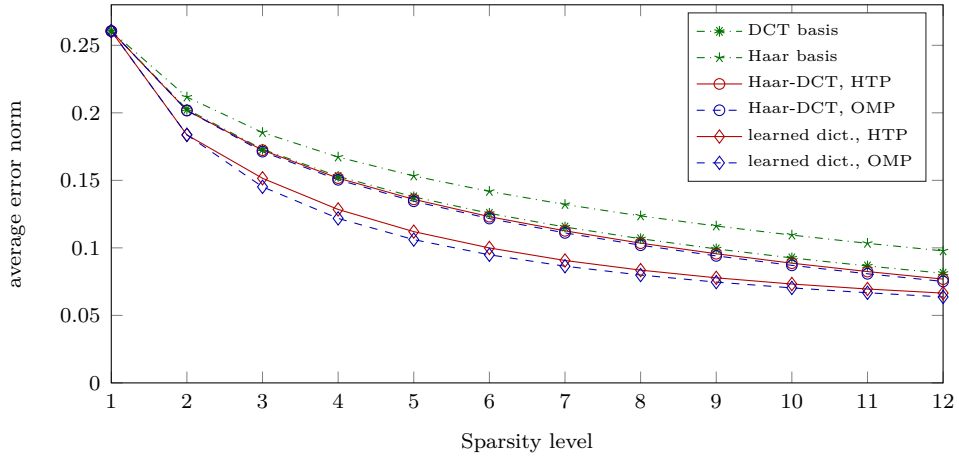


Figure 8: Sparse approximation of image patches (different bases and dictionaries); Sparsity level of the learned dictionary: $5 + 1$

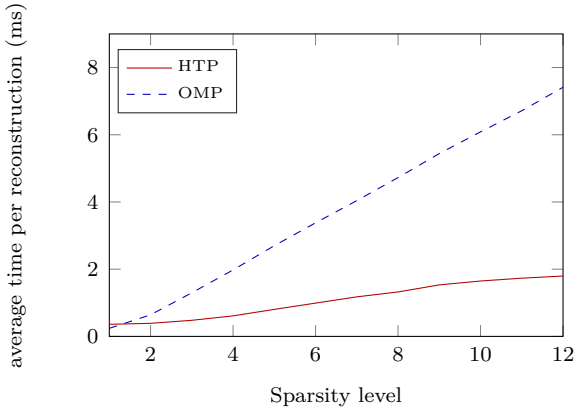


Figure 9: Runtime analysis; Sparsity level of the learned dictionary: $5 + 1$

5 Discussion

This thesis presents a theoretical worst case analysis of the HTP algorithm for sparse approximation and an average case performance assessment through simulation studies comparing HTP to OMP. Additionally to that, the performance of the algorithms is tested with real image data.

The theoretical worst case analysis provides an error estimation for dictionaries that satisfy the incoherence condition $\mu < \frac{1}{\sqrt{3}(3s-1)}$. However, this coherence bound is very strict and in most cases not applicable.

Contrary to the sobering theoretical result, the HTP algorithm shows good performance in simulations with synthetic data when compared to another sparse approximation method, the OMP algorithm. While OMP performs slightly better with input signals generated with non-constant coefficients in the dictionary, HTP's success rates exceed those of OMP for input signals generated with constant coefficients. Furthermore, the HTP algorithm has a much lower average time per reconstruction because it only needs to run a small number of iterations.

When being tested with real image data, OMP's error rates are slightly lower than those of HTP while HTP is still considerably faster. In this image processing example, also a learned dictionary is used, whose random initialization has been modified to fit the image patches. The results of the algorithms with this learned dictionary show a clear advantage over a non-learned dictionary.

The contrast between the poor theoretical worst case result and a good average case performance gives rise to the question whether a theoretical average case analysis could provide insight and more realistic error estimations for the HTP algorithm. Theoretical average case analyses have been done for BP and for thresholding, revealing that on average, BP and thresholding cope well with different model assumptions. Since the HTP algorithm shows good performance in almost all simulation settings considered in this thesis, trying to prove this average performance would be an interesting and promising starting point for further research.

References

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

[2] Blumensath, T., Davies, M. E.: *Iterative thresholding for sparse approximations*, Journal of Fourier Analysis and Applications, 14(5), 629-654, 2008.

[3] Chen, S. S., Donoho, D. L., Saunders, M. A., *Atomic decomposition by basis pursuit*, SIAM Journal on Scientific Computing, 20, 33-61, 1998.

[4] Foucart, S.: *Hard Thresholding Pursuit: An algorithm for compressive sensing*, SIAM Journal on Numerical Analysis 49(6), 2543-2563, 2011.

[5] Foucart, S., Rauhut, H.: *A mathematical introduction to compressive sensing*, Springer SBM, New York, 2013.

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

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

[8] Pati, Y. C., Rezaiifar, R., Krishnaprasad, P. S.: *Orthogonal Matching Pursuit: Recursive function approximation with applications to wavelet decomposition*, Proceedings of the 27th Annual Asilomar Conference on Signals, Systems, and Computers, 40-44, 1993.

[9] Pennebaker, W. B., Mitchell, J. L.: *JPEG Still Image Data Compression Standard*, Springer, New York, 1993.

[10] Schnass K.: *Convergence radius and sample complexity of ITKM algorithms for dictionary learning*, arXiv:1503.07027, 2016. (Preprint)

[11] Schnass K., Vandergheynst P.: *Average performance analysis for thresholding*, IEEE Signal Processing Letters 14(11), 828-831, 2007.

[12] Strang, G.: *The Discrete Cosine Transform*, SIAM Review, 41(1), 135–147, 1999.

[13] Tropp, J. A.: *Greed is Good: Algorithmic results for sparse approximation*, IEEE Transactions on Information Theory 50(10), 2231-2242, 2004.

[14] Tropp, J. A.: *On the conditioning of random submatrices*, Applied and Computational Harmonic Analysis, 25(1), 1-24, 2008.

[15] Welch L. R.: *Lower bounds on the maximum cross correlation of signals*, IEEE Transactions on Information Theory 20, 397-399, 1974.