Face Recognition with Compressive Sensing

Mathias Lohne

Spring, 2017

Abstract

Brief summary of the paper

1 Introduction

Write introduction

In this report we will first give a general introduction to the basic concepts of compressed sensing, and then look at how these techniques can be applied to the face recognition problem.

1.1 Preliminaries and Notation

In this section we will introduce some of the necessary notation and concepts for this report.

Throughout the report we will denote vectors by boldface lower case letters, and matrices by boldface upper case letters. For a vector $\mathbf{x}$, we will by x_i refer to the i'th element in the vector. Similarly, for a matrix $\mathbf{A}$, we will by $\mathbf{a}_i$ refer to the i'th column of $\mathbf{A}$, and by $a_{i,j}$ denote the element on column j and row i.

Sets will be denoted by italic upper case letters, and the cardinality of a set S is denoted as $|S|$. The complement of a set S will be written as $\overline{S}$.

Norms

We begin with the definition of norms.

Definition 1.1. Let V be a vector space over $\mathbb{K}$. A norm $\|\cdot\|$ on V is a function $\|\cdot\| : V \to \mathbb{R}$ such that

(i) $\|\mathbf{v}\| \geq 0$ for all $\mathbf{v} \in V$ with $\|\mathbf{v}\| = \mathbf{0}$ if and only if $\mathbf{v} = \mathbf{0}$

(ii) $\|c\mathbf{v}\| = |c| \, \|\mathbf{v}\|$ for all $\mathbf{v} \in V$ and $c \in \mathbb{K}$

(iii) $\|\mathbf{u}+\mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ for all $\mathbf{u}, \mathbf{v} \in V$

This definition is quite broad, and very general. In this report we will be working more closely with a family of norms called the ℓ_p norms:

Definition 1.2. Let $p \geq 1$ and $p \in \mathbb{R}$. The ℓ_p norm $\|\cdot\|_p$ is defined as

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{\frac{1}{p}}$$

We will not prove here that the ℓ_p-norms actually are norms, but this can be proven for all $p \geq 1$. We observe that for $p = 1$ we get the Manhattan norm, and for $p = 2$ we get euclidean norm. If we let $p \to \infty$ we arrive at the supremum norm. Even though Definition 1.2 does not allow for $p < 1$, if we let $p \to 0$, accept that $0^0 = 0$, and ignore the $1/p$-exponent, we get what is sometimes called the ℓ_0 norm:

Definition 1.3. The ℓ_0 norm $\|\cdot\|_0$ is the number of non-zero entries in $\mathbf{v}$.

It is worth noting that the ℓ_0 norm is strictly speaking not a norm, since $\|\cdot\|_0$ does not fulfill axiom *(ii)* of Definition 1.1. In fact, for all $q < 1$, $\|\cdot\|_q$ is not a norm. Despite this, it is customary to refer to $\|\cdot\|_0$ as the ℓ_0 norm.

Support and Sparsity

Definition 1.4. Let $\mathbf{v} \in \mathbb{C}^N$. The support S of $\mathbf{v}$ is defined as the index set of its non-zero entries, that is:

$$\operatorname{supp} \mathbf{v} = \{j \in \{1, 2, \ldots, N\} \mid v_j \neq 0\}$$

The notion of support yields a new formulation of Definition 1.3: The ℓ_0 norm is simply the cardinality of the support: $\|\mathbf{v}\|_0 = |\operatorname{supp} \mathbf{v}|$.

For a vector $\mathbf{v} \in \mathbb{C}^N$ and a set $S \subset \{1, 2, \ldots, N\}$, we denote by $\mathbf{v}_S$ either the subvector in $\mathbb{C}^{|S|}$ consisting of the entries in $\mathbf{v}$ indexed by S, that is:

$$(\mathbf{v}_S)_i = v_i \text{ for } i \in S \tag{1.1}$$

Or the vector in $\mathbb{C}^N$ which coincides with $\mathbf{v}$ on the indices in S, and is zero otherwise, that is:

$$(\mathbf{v}_S)_i = \begin{cases} v_i & \text{if } i \in S \\ 0 & \text{Otherwise} \end{cases} \tag{1.2}$$

It should always be clear from context which of these is used. Similarly, for a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ we will by $\mathbf{A}_S \in \mathbb{R}^{m \times |S|}$ refer to the matrix consisting of the columns of $\mathbf{A}$ indexed by S.

The final concept we will introduce is the notion of *sparsity*:

Definition 1.5. A vector $\mathbf{v} \in \mathbb{C}^N$ is said to be s-sparse if it has no more than s non-zero entries. That is, $\|\mathbf{v}\|_0 \leq s$

2 A Sparse Introduction to Compressive Sensing

We will introduce the basic idea of compressive sensing with an example quite similar to the one found in [BL13]. Suppose we have 100 coins. We suspect that a few of them might be counterfeit, and thus have a slightly different weight than the normal coins.

The naive approach to finding these coins would be to weigh every one of them with an electronic weight, and detect the coins that are off. In other words, we would have to do as many measurements as there are coins. But what if we weighed more than one coin at a time?

Suppose, to the contrary, that we would include 50 coins in every weighing. The recorded weight would be the sum of all the included coins. Would we be able to make do with less weighings than 100? Say if we for example made 20 measurements, and recorded only the deviation from the expected weight. This would lead to the following system of equations:

$$\mathbf{A}\mathbf{x} = \mathbf{y}$$

Here $\mathbf{A} \in \mathbb{R}^{20\times 100}$ is a matrix where each row corresponds to one weighing, and the elements $a_{i,j}$ is either 0 or 1, depending on whether coin j was included in weighing i or not. The vector $\mathbf{y} \in \mathbb{R}^{20}$ is our measurement vector, and $\mathbf{x} \in \mathbb{R}^{100}$ is the solution to our problem.

Unfortunately, since $\mathbf{A}$ has far more columns than rows, this system is underdetermined, meaning that solving this system the old-fashioned way yields an infinite set of solutions.

Here comes the key idea of compressive sensing: we will assume that our solution vector $\mathbf{x}$ is *sparse*. In our example it would make sense to assume that most coins are not counterfeit, so the solution vector $\mathbf{x}$ (consisting of deviations from the expected weight) would mostly have elements equal to 0. Hence, we want to choose the solution from the solution space of $\mathbf{A}\mathbf{x} = \mathbf{y}$ with the smallest amount of non-zero elements.

The rest of this chapter will consider how to find the sparsest solution in practice, as well some of the properties we need of $\mathbf{A}$ in order to make sure that this procedure works for all s-sparse vectors.

2.1 The General Setting

As we now have seen, if we assume that our solution vector $\mathbf{x}$ is the sparsest solution $\mathbf{z}$ to the system of equations $\mathbf{A}\mathbf{z} = \mathbf{y}$, there is hope to find a unique solution even though our system of equations is underdetermined. In Section 2.2 we will look at when we have unique solutions up to a given sparsity s, but for now we will only consider that minimizing the number of non-zero elements will reduce the solution space drastically. Formally we can write this as an optimization problem as follows:

$$\underset{\mathbf{z}\in\mathbb{C}^N}{\text{minimize}} \ \|\mathbf{z}\|_0 \quad \text{subject to } \mathbf{A}\mathbf{z} = \mathbf{y} \tag{P_0}$$

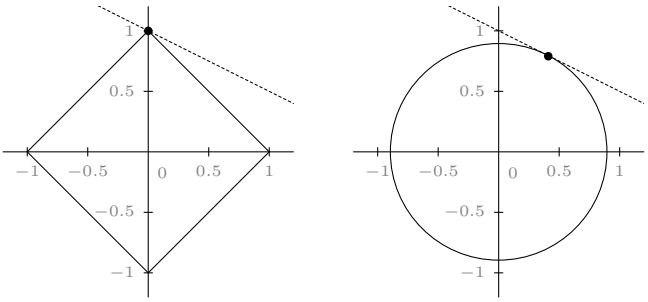

Figure 2.1: Visualization of the two-dimensional case in Example 2.1: ℓ_1 (left) and ℓ_2 (right) balls, along with the solution space to $\mathbf{Az} = \mathbf{y}$ (dashed line)

However, this problem seems to be intractable in practice. It is, in fact, NP-hard in general. A proof of the NP-hardness of (P_0) is found in Section 2.3 of [FR13], and is obtained by reducing (P_0) to the *exact cover by 3-sets* problem, which is known to be NP-complete.

Basis Pursuit

Since (P_0) is computationally hard, we need something to approximate it. One intuitive guess would be to use minimization of another norm, like the ℓ_1 or ℓ_2 norm. It turns out that this is what we usually do in practice.

The question then becomes, which norm do we use? It can be shown that as p gets lower, the ℓ_p norm approximates the ℓ_0 norm better [FR13, Section 4.1]. We will use the lowest value for p possible, while still having a minimization problem that can be solved in polynomial time. Thus, we will use the ℓ_1-norm. Figure 2.1 illustrates why ℓ_1-minimization works well to find the ℓ_0 minimum.

We will illustrate this with an example:

Example 2.1. Let

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad \mathbf{y} = 2$$

We want to find the solution $\mathbf{z}$ to $\mathbf{Az} = \mathbf{y}$ that minimizes the ℓ_0 norm, and we will use the ℓ_1 and ℓ_2 norms to approximate the ℓ_0 norm.

The solution space to $\mathbf{Az} = \mathbf{y}$ yields a line in $\mathbb{R}^2$. Intuitively, we can visualize finding the solution to $\mathbf{Az} = \mathbf{y}$ that minimizes the ℓ_p-norm, as having a ℓ_p-ball $B_p(\mathbf{0}, r)$ centered at the origin, and increasing the radius r until it intersects with this solution space. The intersection is then the optimal solution. This is what we have shown in Figure 2.1.

The result of ℓ_1-minimization gives $\mathbf{z} = (1, 0)$, which is the correct ℓ_0 minimum. The result of ℓ_2-minimization gives $\mathbf{z} \approx (0.39419, 0.8029)$, which is not the correct ℓ_0 minimum.

This example illustrates both why ℓ_1-minimization is a reasonable choice of approximation, and also why ℓ_2-minimization is *not*. We formalize this new problem:

$$\underset{\mathbf{z} \in \mathbb{C}^N}{\text{minimize}} \ \|\mathbf{z}\|_1 \quad \text{subject to } \mathbf{Az} = \mathbf{y} \tag{P_1}$$

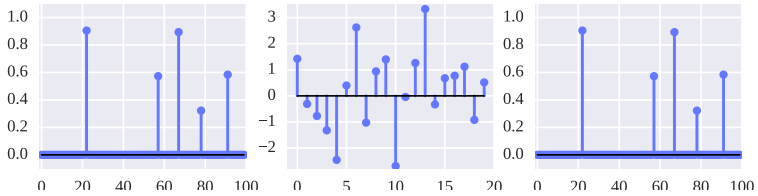

Figure 2.2: *Left*: The original 5-sparse vector $x \in \mathbb{R}^{100}$. *Center*: The sensed vector $y \in \mathbb{R}^{20}$. *Right*: Result of ℓ_1-minimization

This is also called *basis pursuit*. We will later discuss how one can guarantee that the solution to (P_1) is actually the solution to (P_0).

An illustration of this procedure is found in Figure 2.2. In this example we have drawn a random 5-sparse vector $\mathbf{x} \in \mathbb{R}^{100}$. Using a random sensing matrix $\mathbf{A} \in \mathbb{R}^{20\times 100}$ where the elements $a_{i,j} \overset{iid}{\sim} N(0,1)$ for all i, j, we obtain our sampled vector $\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathbb{R}^{20}$. We have then applied Vegard Antun's implementation of the *Orthogonal Matching Pursuit* described in [FR13, Section 3.2] to solve (P_1), giving us our reconstructed vector.

Because we have used a random sensing matrix, there is no guarantee that basis pursuit works. We only have a certain probability that the reconstruction yields the correct result. Running the experiment described above several times actually gives the wrong result some times. This is because there is only a certain probability that our random matrix $\mathbf{A}$ exhibits properties such as the Null Space Property, which is necessary in order for basis pursuit to work.

Recasting (P_1) as a Linear Program

Now that we have a tractable way of finding sparse solutions, we will look at one way to actually solve (P_1). In this section we will see how one can use linear programming to do this. Linear programming (LP) is the study of problems on the form

$$\underset{\mathbf{x}\in\mathbb{K}^N}{\text{maximize}}\ \mathbf{c}^T\mathbf{x} \qquad \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}\,,\ \ \mathbf{x} \geq \mathbf{0} \tag{2.1}$$

Thus, to use one of the algorithms developed for LP, we need to rewrite (P_1) as a linear program (ie, on the form (2.1)).

In order to do this, three problems arise. Fist, we see from Equation (2.1) that LP problems do not take constraints on equality form, which is what we have in (P_1). Second, LP problems only minimizes linear functions (ie, functions that can be written as a dot product between the solution vector and some constant vector). Absolute values, as we have in the ℓ_1 norm, can not be described this way. Third, a general LP problem requires all values in $\mathbf{x}$ be non-negative. This is not a constraint we have in (P_1).

We begin by addressing the first issue. This can be quite easily solved by observing that an equality constraint can be rewritten as two inequality constraints like so:

$$\mathbf{A}\mathbf{z} = \mathbf{y} \iff \mathbf{A}\mathbf{z} \leq \mathbf{y} \text{ and } \mathbf{A}\mathbf{z} \geq \mathbf{y}$$

The second issue is a common one in LP. Thus there are also common ways to work around it. One popular way of doing this is to introduce new variables t_i such that $|z_i| \leq t_i$ for $i \in \{1, 2, \ldots, N\}$. Thus,

$$\underset{\mathbf{z} \in \mathbb{C}^N}{\text{minimize}} \sum_{i=1}^{N} |z_i|$$

can be rewritten as

$$\underset{\mathbf{z}, \mathbf{t} \in \mathbb{C}^N}{\text{minimize}} \sum_{i=1}^{N} t_i \qquad \text{subject to } |z_i| \leq t_i \text{ for all } i \in \{1, 2, \ldots, N\}$$

we can then rewrite the constraints as

$$\begin{aligned} z_i - t_i &\leq 0 \\ -z_i - t_i &\leq 0 \end{aligned} \qquad \text{for all } i \in \{1, 2, \ldots, N\}$$

to arrive at the standard LP form. The t_i's are clearly non-negative since they are defined to be greater than or equal to an absolute value.

The third issue is also a quite common one in LP. We now have two decision vectors, $\mathbf{z}$ and $\mathbf{t}$, where the elements of $\mathbf{z}$ does not need to be non-negative. We will solve this by introducing two new decision vectors $\mathbf{z}_+$ and $\mathbf{z}_-$, which will replace $\mathbf{z}$ in the problem formulation.

We define $\mathbf{z}_+$ and $\mathbf{z}_-$ as follows:

$$(\mathbf{z}_+)_i = \begin{cases} z_i & \text{if } z_i > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (\mathbf{z}_-)_i = \begin{cases} -z_i & \text{if } z_i < 0 \\ 0 & \text{otherwise} \end{cases}$$

It is clear that $\mathbf{z} = \mathbf{z}_+ - \mathbf{z}_-$, it is also clear that $\mathbf{z}_+, \mathbf{z}_- \geq \mathbf{0}$. Substituting in $\mathbf{z}_+ - \mathbf{z}_-$ for $\mathbf{z}$ in the original problem will then solve this issue.

A final issue is that general LP problems concerns maximization problems, however (P_1) is a minimization problem. This is easily solved by observing that minimizing a function f is equivalent to maximizing $-f$.

Combining all of the above we arrive at the LP formulation of basis pursuit:

$$\begin{aligned} \text{maximize} \quad & -\textstyle\sum_{i=1}^{n} t_i \\ \text{subject to} \quad & \mathbf{A}\mathbf{z}_+ - \mathbf{A}\mathbf{z}_- && \leq \mathbf{y} \\ & -\mathbf{A}\mathbf{z}_+ + \mathbf{A}\mathbf{z}_- && \leq -\mathbf{y} \\ & \mathbf{z}_+ - \mathbf{z}_- - \mathbf{t} && \leq \mathbf{0} \\ & -\mathbf{z}_+ + \mathbf{z}_- - \mathbf{t} && \leq \mathbf{0} \\ & \mathbf{z}_+, \mathbf{z}_-, \mathbf{t} && \geq \mathbf{0} \end{aligned} \tag{2.2}$$

Thus, we can use all the celebrated algorithms for Linear Programming to solve (P_1), such as the Simplex method, or various interior point methods [Van14].

2.2 Good Sensing Matrices

In this section we will look at the minimum number of measurements required, as well as some of the features we want our sensing matrix $\mathbf{A}$ to possess, in order to ensure that basis pursuit works well.

We will start by looking at how we can ensure unique s-sparse solutions to the ℓ_1-minimization. Then we will look at how we can make sure that the solution to (P_1), which is the problem we will solve in practice, is in fact the solution to (P_0), which is the problem we actually want to solve. Finally we will look at which matrices will perform well with the different recovering algorithms used.

Minimum Number of Measurements

We begin by looking at how many measurements we need in order to ensure that (P_0) has only one s-sparse solution. We begin by stating the main result of this section:

Theorem 2.2. *If $\{\mathbf{z} \in \mathbb{C}^N \mid \mathbf{Az} = \mathbf{Ax}, \|\mathbf{z}\|_0 \leq s\} = \{\mathbf{x}\}$, that is, $\mathbf{x}$ is the unique s-sparse solution to (P_0), then the number of measurements m must satisfy $m \geq 2s$.*

Before proving this theorem, we need the following lemma:

Lemma 2.3. *Every s-sparse vector $\mathbf{x}$ is the unique s-sparse solution to (P_0) with $\mathbf{y} = \mathbf{Ax}$ if and only if every set of $2s$ columns of $\mathbf{A}$ is linearly independent.*

We will not prove Lemma 2.3 in this report, but a proof can be found in [FR13, Theorem 2.13]. We are now ready to prove the main result:

Proof of Theorem 2.2. Assume that it is possible to uniquely recover any s-sparse vector $\mathbf{x}$ from the knowledge of its measurement vector $\mathbf{y} = \mathbf{Ax}$. Then, by Lemma 2.3, we have that every set of $2s$ columns of $\mathbf{A}$ must be linearly independent. This implies that $\operatorname{rank} \mathbf{A} \geq 2s$. From elementary linear algebra we know that the rank of a matrix can not be bigger that the number of rows, hence $\operatorname{rank} \mathbf{A} \leq m$. Combining this, we get that

$$2s \leq \operatorname{rank} \mathbf{A} \leq m$$

which concludes the proof. □

A fra fouriermat

The Null Space Property

So far, we have only looked at the intuitive reasoning of why ℓ_1-minimization works well to find the ℓ_0-minimum. In this section we will formalize this relationship using the *Null Space Property* (often abbreviated NSP). It can be shown that, for the NSP, the real and complex case are equivalent (for a formal statement and proof, see Theorem 4.7 in [FR13]). Hence we will state the definitions and results for a field $\mathbb{K}$, which can be either $\mathbb{R}$ or $\mathbb{C}$. The NSP is defined as follows:

Definition 2.4. A matrix $\mathbf{A} \in \mathbb{K}^{m \times N}$ is said to satisfy the *null space property* relative to a set $S \subset \{1, 2, \ldots, N\}$ if

$$\|\mathbf{v}_S\|_1 < \|\mathbf{v}_{\overline{S}}\|_1 \quad \forall\, \mathbf{v} \in \ker \mathbf{A} \setminus \{\mathbf{0}\}$$

It is said to satisfy the null space property of order s if it satisfies the null space property relative to any set $S \subset \{1, 2, \ldots, N\}$ with $|S| \leq s$

The definition of the NSP might seem a bit arbitrary, but as we will soon see, the NSP is directly connected to the success of basis pursuit.

Theorem 2.5. *Given a matrix $\mathbf{A} \in \mathbb{K}^{m \times N}$, every vector $\mathbf{x} \in \mathbb{K}^N$ supported on a set S is the unique solution to* (P_1) *with $\mathbf{y} = \mathbf{A}\mathbf{x}$ if and only if $\mathbf{A}$ satisfies the NSP relative to S.*

Proof. We will begin by proving that if a vector $\mathbf{x}$ supported on S uniquely solves (P_1), then $\mathbf{A}$ satisfies the NSP relative to S.

Given an index set S, assume that every vector $\mathbf{x} \in \mathbb{K}^N$ supported on S is the unique solution to

$$\underset{\mathbf{z} \in \mathbb{C}^N}{\text{minimize}} \, \|\mathbf{z}\|_1 \quad \text{subject to } \mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{x} \tag{P_1}$$

Since $\ker \mathbf{A}$ is a subspace of $\mathbb{K}^N$, it is clear that for any $\mathbf{v} \in \ker \mathbf{A} \setminus \{0\}$, the vector $\mathbf{v}_S$ is the unique solution to

$$\underset{\mathbf{z} \in \mathbb{C}^N}{\text{minimize}} \, \|\mathbf{z}\|_1 \quad \text{subject to } \mathbf{A}\mathbf{z} = \mathbf{A}\mathbf{v}_S \tag{2.3}$$

Because $\mathbf{v} \in \ker \mathbf{A}$, we have that $\mathbf{A}\mathbf{v} = \mathbf{0}$, which means that $\mathbf{A}(\mathbf{v}_S + \mathbf{v}_{\overline{S}}) = \mathbf{0}$, giving us that $\mathbf{A}(-\mathbf{v}_{\overline{S}}) = \mathbf{A}\mathbf{v}_S$. Hence it clear that $-\mathbf{v}_{\overline{S}}$ is also a feasible solution to (2.3), but since $\mathbf{v}_S$ is assumed to be the *unique* optimal solution to (2.3), we get that $\|\mathbf{v}_S\|_1 < \|-\mathbf{v}_{\overline{S}}\|_1$. Since $\|\cdot\|_1$ is a norm, we have that $\|-\mathbf{v}_{\overline{S}}\|_1 = |-1| \, \|\mathbf{v}_{\overline{S}}\|_1 = \|\mathbf{v}_{\overline{S}}\|_1$ from Definition 1.1. Thus, we arrive at the following inequality:

$$\|\mathbf{v}_S\|_1 < \|\mathbf{v}_{\overline{S}}\|_1$$

This establishes the NSP for $\mathbf{A}$, relative to S.

To prove the other implication, assume first that the NSP holds for $\mathbf{A}$, relative to a given set S. Let $\mathbf{x}$ be a vector in $\mathbb{K}^N$ supported on S. Let $\mathbf{z} \in \mathbb{K}^N$ be a vector that satisfies $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{z}$, and assume that $\mathbf{x} \neq \mathbf{z}$. Our goal will be to show that $\|\mathbf{z}\|_1$ must be strictly bigger than $\|\mathbf{x}\|_1$, which will prove the uniqueness of the solution.

Define $\mathbf{v} = \mathbf{x} - \mathbf{z}$. Since $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{z}$, we have that

$$\mathbf{0} = \mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{z} = \mathbf{A}(\mathbf{x} - \mathbf{z}) = \mathbf{A}\mathbf{v}$$

This means that $\mathbf{v} \in \ker \mathbf{A}$. Since $\mathbf{x} \neq \mathbf{z}$, we also have that $\mathbf{v} \neq \mathbf{0}$. If we use the triangle inequality of norms, as well as the definition of $\mathbf{v}$, we obtain

$$\|\mathbf{x}\|_1 = \|\mathbf{x} - \mathbf{z}_S + \mathbf{z}_S\|_1 \leq \|\mathbf{x} - \mathbf{z}_S\|_1 + \|\mathbf{z}_S\|_1 = \|\mathbf{v}_S\|_1 + \|\mathbf{z}_S\|_1$$

Now, using the assumption that $\mathbf{A}$ satisfies the NSP relative to S we get the next inequality

$$\|\mathbf{v}_S\|_1 + \|\mathbf{z}_S\|_1 < \|\mathbf{v}_{\overline{S}}\|_1 + \|\mathbf{z}_S\|_1$$

Using the definition of $\mathbf{v}$ and $\mathbf{z}$ again, we arrive at our final result:

$$\|\mathbf{v}_{\overline{S}}\|_1 + \|\mathbf{z}_S\|_1 = \|\mathbf{x}_{\overline{S}} - \mathbf{z}_{\overline{S}}\|_1 + \|\mathbf{z}_S\|_1 = \|-\mathbf{z}_{\overline{S}}\|_1 + \|\mathbf{z}_S\|_1 = \|\mathbf{z}\|_1$$

This proves that $\|\mathbf{x}\|_1 < \|\mathbf{z}\|_1$ for any $\mathbf{z} \in \mathbb{K}^N$ satisfying $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{z}$ and $\mathbf{x} \neq \mathbf{z}$. This establishes the required minimality of $\|\mathbf{x}\|_1$, and thus the uniqueness of the solution. □

Theorem 2.5 is not that interesting by itself, but if we let the set S vary, it immediately yields a more general result:

Corollary 2.6. *Given a matrix $\mathbf{A} \in \mathbb{K}^{m\times N}$, every s-sparse vector $\mathbf{x} \in \mathbb{K}^N$ is the unique solution to (P_1) with $\mathbf{y} = \mathbf{A}\mathbf{x}$ if and only if $\mathbf{A}$ satisfies the NSP of order s.*

Before we prove this result, we will give a small remark: Corollary 2.6 shows that if $\mathbf{A}$ satisfies the NSP of order s, the ℓ_1-minimization strategy of (P_1) will actually solve (P_0) for all s-sparse vectors.

Proof of Corollary 2.6. Assume every s-sparse vector $\mathbf{x} \in \mathbb{K}^N$ is the unique solution to (P_1). Then, for every set S with $|S| \leq s$ we can find a vector $\mathbf{x}' \in \mathbb{K}$ supported on S which is the unique solution to (P_1). By Theorem 2.5 we then have that $\mathbf{A}$ must satisfy the NSP relative to S. Since this is true for all S with $|S| \leq s$, $\mathbf{A}$ must satisfy the NSP of order s.

Conversely, assume that $\mathbf{A}$ satisfies the NSP of order s. Then, from Definition 2.4, for every set S with $|S| \leq s$ we have that $\mathbf{A}$ satisfies the NSP relative to S. From Theorem 2.5 we have that a vector $\mathbf{x} \in \mathbb{K}^N$ is supported on S only if it is the unique solution to (P_1). Since this is true for any set S with $|S| \leq s$, it is true for any s-sparse vector. □

Coherence

We want our recovering of $\mathbf{x}$ to be fast and reliable. Since the i'th element of our measurement vector $\mathbf{y}$ is the inner product between the row i of $\mathbf{A}$ and $\mathbf{x}$, it makes intuitive sense that the columns of $\mathbf{A}$ should be as "scattered" as possible. This leads us to the definition of coherence:

Definition 2.7. Let $\mathbf{A} \in \mathbb{C}^{m\times N}$ be a matrix with ℓ_2-normalized columns $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_N$. The *coherence* μ of $\mathbf{A}$ is defined as

$$\mu = \max_{1\leq i\neq j\leq N} |\langle \mathbf{a}_i, \mathbf{a}_j \rangle|$$

It is worth noting that the coherence of a matrix $\mathbf{A}$ is 0 if and only if $\mathbf{A}$ is orthogonal. This agrees well with the intuitive approach discussed above. However, since compressive sensing deals with situations where $m < N$, this will never be the case for any of the matrices we will look at.

The question now becomes, how low can the coherence get? We know that we will never reach a coherence of 0, but how close can it be? The next lemma gives us a lower bound on the coherence:

Lemma 2.8. *The coherence μ of a matrix $\mathbf{A} \in \mathbb{K}^{m\times N}$ with ℓ_2-normalized columns satisfies the following inequality:*

$$\mu \geq \sqrt{\frac{N-m}{m(N-1)}}$$

This bound is often called the *Welch bound*. The proof for this Lemma is rather technical, and will therefore be omitted in this report. A complete proof for Lemma 2.8 can be found in [FR13, Theorem 5.7].

Eksempel med matriser med lav koherens?

2.3 Sparsity in the Real World

Compressibility

The general framework developed in Section 2.2 is designed for *sparse* vectors without any errors. What if our signal is almost sparse, but not quite? Or what if the sampled signal is slightly distorted? In this section we will briefly cover how we can extend the theory presented in Section 2.2 to cover such cases.

First, we will introduce the notion of compressibility. A compressible vector is a vector where most entries is *almost* zero. To more precisely define a compressible vector, we must first define what we mean by almost zero:

Definition 2.9. For any $p > 0$, the ℓ_p-error of best s-term approximation to a vector $\mathbf{x} \in \mathbb{C}^N$ is defined by $\sigma_s(\mathbf{x})_p = \inf\{\|\mathbf{x} - \mathbf{z}\|_p : \mathbf{z} \text{ is } s\text{-sparse}\}$

We say that a vector $\mathbf{x}$ is s-compressible if $\sigma_s(\mathbf{x})_1$ is "small". The second potential issue we will cover is when our measured vector $\mathbf{y}$ contains some noise, such that $\mathbf{Ax} \approx \mathbf{y}$. If the distance between the distorted measurements $\mathbf{y}$ and the real, unbiased signal $\mathbf{Ax}$ is bounded by a parameter ε, we can rewrite the approximation constraint as $\|\mathbf{Ax}-\mathbf{y}\|_2 < \varepsilon$. This motivates the following variant of basis pursuit:

$$\underset{\mathbf{z}\in\mathbb{C}^N}{\text{minimize}} \qquad \text{subject to } \|\mathbf{Az} - \mathbf{y}\|_2 \leq \varepsilon \tag{$\mathrm{P}_{1,\varepsilon}$}$$

It can be shown that if the sensing matrix exhibits a strengthening of the NSP called *robust NSP*, the error made by solving $(\mathrm{P}_{1,\varepsilon})$ is bounded by a weighted sum of $\sigma_s(\mathbf{x})_1$ and ε [FR13, Section 4.3].

Achieving Sparsity or Compressibility

So far we have just assumed our solution vector $\mathbf{x}$ to be sparse or compressible. However, most real life signals are rarely sparse. Images are usually not mostly black, and songs are usually not mostly silence. Hence, we need some way to represent natural signals in a sparse way.

We will achieve this by applying what is known as a *sparsifying transform*. Many such transforms exist, but in this report we will consider the *Haar wavelet transform* as an example. The Haar wavelet is by far not the most efficient sparsifying transform, but it is quite understandable, which is why we have chosen it.

The key idea in a wavelet transform is to take some object, expressed in a high resolution wavelet basis, and express it in terms of a lower resolution basis, and a detail basis. In the specific case of the Haar wavelet, those functions are defined as follows:

$$\phi(t) = \begin{cases} 1 & \text{if } 0 \leq t < 0 \\ 0 & \text{otherwise} \end{cases} \qquad \psi(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1/2 \\ -1 & \text{if } 1/2 \leq t < 1 \\ 0 & \text{otherwise} \end{cases}$$

Figure 2.3: *Left:* the original image of Lily. *Right:* the image after a 1-level discrete wavelet transform using the Haar wavelet. In this example, we have used the 2D DWT implementation provided in [Rya16].

By shifting and scaling those functions we get a basis for the low resolution space (from the ϕ's) and the detail space (from the ψ's). Since a digital image is piecewise linear in every pixel, it is clear that a digital image of size $n \times n$ is in the $n \times n$ dimensional high resolution space for the Haar wavelet.

Overgang til diskret tilfelle

The Haar Discrete Wavelet Transform (Haar DWT) is essentially a change of coordinates from the higher resolution wavelet basis basis, to a lower resolution and detail basis. Figure 2.3 illustrates what happens when the DWT is applied to an image. The upper left corner of the resulting image is the low resolution space. This is usually not any sparser or more compressible than the original image. However, in the upper right, lower left and lower right corners we see the detail space. This is highly compressible, and it is also clear that the total compressibility of the image has increased (ie, number of non-zero components has decreased). We will denote the corresponding matrix to this change of coordinates by $\mathbf{\Psi}$.

We could assume that this change of coordinates have been done prior to the sensing, so that our solution vector $\mathbf{x} = \mathbf{\Psi}\mathbf{x}'$ is sparse. Here $\mathbf{x}'$ is the underlying non-sparse solution. However, we will instead include the change of coordinates in our sensing matrix $\mathbf{A}$. Hence the sensing matrix becomes the following product:

$$\mathbf{A} = \mathbf{P}_\Omega \mathbf{\Phi} \mathbf{\Psi}^{-1} \tag{2.4}$$

Here, $\mathbf{P}_\Omega$ is a matrix describing the down-sampling, $\mathbf{\Phi}$ denotes the sampling pattern used, and $\mathbf{\Psi}$ is the basis in which $\mathbf{x}$ is sparse. Typical choices include letting $\mathbf{\Psi}$ be some wavelet basis, letting $\mathbf{\Phi}$ be rows from the discrete Fourier matrix, and letting $\mathbf{P}_\Omega$ pick out the first n rows, or n randomly chosen rows.

?

This means that solving (P_1) results in a recovered vector $\mathbf{z}$ also expressed in the $\mathbf{\Psi}$-basis. To recover our real non-sparse vector $\mathbf{z}'$ we would then have to apply the reversed change of coordinates:

$$\mathbf{z}' = \mathbf{\Psi}^{-1}\mathbf{z}$$

For ease of notation, we will simply refer to the overall matrix as $\mathbf{A}$, even though we think of it as a product of multiple matrices. The concepts we discussed earlier in Section 2.2 will apply to this product $\mathbf{A}$. We will also note that we will later use $\mathbf{\Phi}$ to denote a matrix generated from training images. This is a different matrix, and should not be confused with the sampling pattern here.

3 Applications to Face Recognition

We will now shift our focus away from general compressive sensing, and look at how these techniques of sparse recovery can be used to build a framework for a complete face recognition system.

The face recognition problem is a classical one in the area of statistical learning and machine intelligence. Hence it is a broadly studied problem, with many proposed solutions. The classical way to do face recognition is to first do what is called feature extraction. One can think of feature extraction as a projection to a lower dimensional feature space, such that the requirements for memory and computational power is reduced. Popular methods for this includes Principal Component Analysis and Discrete Cosine Transform [BW14].

After projecting the images down to a lower resolution space, one typically applies some statistical classification scheme. Typical classifiers used in classical face recognition include Linear Discriminant Analysis (LDA) [BW14], K Nearest Neighbor/Subspace (KNN/KNS) [LHK05] and (Linear-kernel) Support Vector Machine (SVM) [Wri+09]. These methods seem to work well when in a controlled environment. However, when parameters such as lighting or noise level changes, or when a subject has occluded parts of his/her face (like the addition of glasses), these methods often begin to struggle [EK12].

In this chapter we will introduce a new classification scheme called Sparse Representation-based Classification (SRC) [Wri+09]. This is an alternative method to the LDA or KNN discussed above, but does not replace the dimensionality reduction. In this report, however, we will not discuss feature extraction in any more detail, but focus on the classification scheme.

In order to do face recognition, we first need a set of N labeled *training images* $\{(\phi_i, l_i)\}_{i=1}^N$. This is a set of images which our algorithm will use to learn what the different people we want to recognize looks like. Here, (ϕ_i, l_i) denotes the tuple consisting of the i'th image ϕ_i, and a label $l_i \in \{1, 2, \dots, C\}$ indicating which of the C subjects the image ϕ_i is of. The task og the system is then, given a new test image $\mathbf{y}$, to say which of the C subjects is pictured in $\mathbf{y}$.

3.1 Sparse Representation-based Classification

At first we must address a potential problem. All of the theory developed in Chapter 2 concerns vectors in $\mathbb{K}^N$, while we usually think of images as matrices in $\mathbb{R}^{m \times n}$. It seems we have a problem with dimensionality. However this problem can be easily solved by simply stacking all the columns of the image matrix $\mathbf{A}$ on

top of each other in the following way:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \end{bmatrix} \in \mathbb{R}^{m \times n} \quad \text{becomes} \quad \mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} \in \mathbb{R}^{mn}$$

We begin by making an observation: if our training images of subject i is of varying illumination, and if we assume that they are all aligned correctly, we will expect a test image of subject i to be closely approximated by a linear combination of the test images. That is, for a new test image $\mathbf{y}$ of subject i, there exists k_i coefficients $c_1, c_2, \ldots, c_{k_i} \in \mathbb{R}$ (here, k_i is the number of images of subject i) such that:

$$\mathbf{y} \approx \sum_{j \mid l_j = i} \phi_j c_j \tag{3.1}$$

We note that if we define $\mathbf{c}_i \in \mathbb{R}^{k_i}$ to be the vector of all the coefficients, and $\mathbf{\Phi}_i \in \mathbb{R}^{mn \times k_i}$ to be the matrix consisting of all the corresponding images as columns, we can rewrite (3.1) as:

$$\mathbf{y} \approx \mathbf{\Phi}_i \mathbf{c}_i \tag{3.2}$$

This is of course assuming that i is known, which of course it is not. However this is easily solved by padding $\mathbf{c}_i$ with zeros. This is safe to do, since we expect all the other coefficients to be 0 anyway (at least close to 0). Thus we arrive at the zero-padded version of $\mathbf{c}_i$:

Forenkle?

$$\mathbf{c}_0 = \begin{bmatrix} \cdots & \mathbf{0}^T & \mathbf{c}_i^T & \mathbf{0}^T & \cdots \end{bmatrix}^T \in \mathbb{R}^N$$

Now, if we concatenate all the images in the entire training database into a matrix as such:

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{\Phi}_2 & \cdots & \mathbf{\Phi}_C \end{bmatrix} \in \mathbb{R}^{mn \times N}$$

where $\mathbf{\Phi}_i$ is the matrix consisting of all the training images of subject i concatenated, we see that we can rewrite (3.2) as:

$$\mathbf{y} \approx \mathbf{\Phi} \mathbf{c}_0 \tag{3.3}$$

This will clearly be the case, no matter what i is.

Summarizing what we have shown so far, we get that we are given a sensed testing image $\mathbf{y}$, and we are trying to find a vector of coefficients $\mathbf{c}_0$ such that $\mathbf{y} \approx \mathbf{\Phi} \mathbf{c}_0$. We also know that $\mathbf{c}_0$ should be highly sparse, since we expect most images in the database to *not* be similar to the testing image. In fact, we only expect an average of $1/C$ entries in $\mathbf{c}_0$ to be non-zero.

In the last chapter we introduced theory for finding the sparsest solution to such systems of equations. However, we now have a system of approximations. If we assume that the error is less than some error term ε, we can state a minimization problem that we expect $\mathbf{c}_0$ to be the optimal solution to:

$$\underset{\mathbf{c} \in \mathbb{R}^N}{\text{minimize}} \ \|\mathbf{c}\|_1 \qquad \text{subject to } \|\mathbf{y} - \mathbf{\Phi}\mathbf{c}\|_2 \leq \varepsilon \tag{3.4}$$

Algorithm 3.1 Sparse Representation-based Classification for face recognition

Input: A matrix of training images $\mathbf{\Phi} \in \mathbb{R}^{mn \times N}$ of C persons, a test image $\mathbf{y} \in \mathbb{R}^N$, and a tolerance $\varepsilon > 0$.

1. Normalize the columns of $\mathbf{\Phi}$ to have a ℓ_2 norm of 1
2. Solve the ℓ_1-minimization problem:

$$\hat{\mathbf{c}}_0 = \arg\min_{\mathbf{c}} \|\mathbf{c}\|_1 \quad \text{subject to } \|\mathbf{\Phi c} - \mathbf{y}\| \leq \varepsilon$$

3. Compute the residuals $r_i(\mathbf{y}) = \|\mathbf{y} - \mathbf{\Phi}_i(\hat{\mathbf{c}}_0)_i\|_2$ for all $i \in \{1, 2, \ldots, C\}$. Here $\mathbf{\Phi}_i$ and $(\hat{\mathbf{c}}_0)_i$ denotes all the training images and estimated coefficients associated with person i.

Output: $\hat{l}_i$ = identity of $\mathbf{y} = \arg\min_i r_i(\mathbf{y})$

We observe that (3.4) has the same form as ($\mathrm{P}_{1,\varepsilon}$). Thus, we can use compressive sensing techniques to solve this minimization problem.

By finding an optimal solution to (3.4), we get a vector $\hat{\mathbf{c}}_0$ of estimated coefficients. It remains to determine which person we believe the image to be of. We will do this by choosing the person who minimizes the squared error term. That is, the person i who minimizes the distance between the sampled test image $\mathbf{y}$ and the linear combination of training images of person i. In this respect, SRC can be thought of as a type of Nearest Neighbor.

Summarizing all of the above, we get the algorithm described in Algorithm 3.1.

3.2 Addressing Corruption and Occlusion

A common problem when dealing with numerical problems of any sort is the introduction of round-off errors. A typical computer today uses 64 bit processors, this means that the processor (and all the other components) can store and process 64 binary digits of a number. Any real number must then be approximated by the closest number possible to write with 64 binary digits. Typically, this introduces a small error $\varepsilon_i < 2^{-53} \approx 10^{-16}$ for every pixel in the image. In addition to round-off errors, all image sensors introduce some image noise.

In addition to uniform noise, another problem we have to address is occlusion. A common problem with many of the classical face recognition systems is their incapability of dealing with occlusion. If a subject puts on a pair of sunglasses, let his/hers hair grow, cuts his beard, or so forth, an ideal system should still be able to classify the subject correctly.

Definition 3.1. We say a sequence of error-signal $(\mathbf{x}, \mathbf{e})$ exhibits *proportional growth* with parameters $\delta > 0$, $\rho \in (0, 1)$, $\alpha > 0$ if

$$n = \lfloor \delta m \rfloor, \quad \|\mathbf{e}_0\|_0 = \lfloor \rho m \rfloor, \quad \|\mathbf{x}_0\|_0 = \lfloor \alpha m \rfloor$$

Before we state the main result of this section, we will give an interpretation of the parameters involved in Definition 3.1. Our first parameter, n, gives a ratio between the length of the real signal, and the length of the measured signal. In other words, n, is the compression rate. The next two parameters deals with sparsity: α is a measure of the sparsity of the uncorrupted signal, and ρ is the sparsity of the error.

If a signal exhibits this property, we have that the following result holds:

Theorem 3.2. *Fix any $\delta > 0$, $\rho < 1$. Suppose that $\mathbf{A}$ is a random matrix with columns drawn independently from a multivariate normal distribution as such:*

$$\mathbf{a}_i \overset{iid}{\sim} N\left(\boldsymbol{\mu}, \frac{\nu^2}{m}\mathbf{I}_m\right), \quad \|\boldsymbol{\mu}\|_2 = 1, \quad \|\boldsymbol{\mu}\|_\infty \leq C_\mu m^{-1/2}$$

for some constant C_μ. Assume that ν is sufficiently small, that $J \subset \{1, 2, \ldots, m\}$ is a uniform random subset of size ρm, that $\boldsymbol{\sigma} \in \mathbb{R}^m$ with $\boldsymbol{\sigma}_J \overset{iid}{\sim}$ Unif$(\{-1, 1\})$ (independent of J), that $\boldsymbol{\sigma}_{\overline{J}} = \mathbf{0}$ and that m is sufficiently large. Then with probability at least $1 - Ce^{-\varepsilon^* m}$ in $A, J, \boldsymbol{\sigma}$, for all $\mathbf{x}_0$ with $\|\mathbf{x}_0\|_0 \leq \alpha^* m$ and any $\mathbf{e}_0$ with signs and support $(\boldsymbol{\sigma}, J)$,

$$(\mathbf{x}_0, \mathbf{e}_0) = \underset{(\mathbf{x},\mathbf{e})}{\arg\min} \|\mathbf{x}\|_1 + \|\mathbf{e}\|_1 \qquad \text{subject to } \mathbf{A}\mathbf{x} + \mathbf{e} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}_0 \tag{3.5}$$

and the minimizer is uniquely defined.

A proof of this theorem will not be given here, but is found in [WM10] (and it is very technical). We will instead give a small interpretation of its consequences.

It is worth noting that many of the technical assumptions in Theorem 3.2 are present only to make the theorem provable. The main result is that we can recover a signal by using ℓ_1-minimization, even if the signal has some noise, and even if the noise is not sparse. However, for this to work we need our error and signal to exhibit certain properties, and the most important one is the proportional growth.

Intuitively, we can think of the result this way: If our error is very dense, we need the signal to be very sparse, and if our signal is dense, we need a sparse error in order for the recovering scheme to work.

One last problem we will not look into is misalignment. Usually, images of faces are not perfectly aligned and cropped, so a fully functioning face recognition system should be able to deal with misaligned images. A more in-depth look at this can be found in Section 12.5 of [EK12].

4 Practical Implementation

We now shift our attention away from the purely theoretical aspect, and towards the practical side. In this chapter we will seek to answer two main questions: How can a face recognition system using the SRC described in Section 3.1 be implemented? And how does it perform compared to the classical, state-of-the-art methods? We will begin by addressing the latter, and then look at some problems that arise when implementing the system.

In order to implement and test a face recognition system, one will need a database of faces to test on. Many such databases exists, and some include the CMU multi-pie, the Yale database (and its successor, the extended Yale B database) and the AR Face Database.

4.1 Comparison with Classical Methods

Now that we have introduced a new classifier for face recognition, an immediate question becomes: Why develop a new classification scheme based on compressive sensing when there already exists multiple statistical classifiers suitable for face recognition?

In the field of statistical and machine learning, one studies, among other things, different classifiers, such as the LDA, KNN or SVM discussed earlier. Typically, we have that for one specific application, different classifiers will perform differently. Thus, for a given application, we can rate how the different classifiers perform. A common metric for classification performance is to observe the *test error rate*.

In order to precisely test the accuracy of a classification model, we need some testing data independent from the training data. Typically, this is achieved by dividing the total data available into two sets: a training set used to build the model, and a test set used to test the model. For our case, we will split our set of images $\{(\phi_i, l_i)\}_{i=1}^N$ in two, and make sure that for each person i, we have equally many images of i in the training and the testing set. In the rest of this section, we will by $I \subset \{1, 2, \ldots, N\}$ denote the set of indices for the training images, and by $J = \{1, 2, \ldots, N\} \setminus I$ denote the set of indexes for the testing images.

Given a model and a set of testing images, the *test error rate* is given as

$$\frac{1}{|J|} \sum_{i \in J} \mathcal{I}(\hat{l}_i \neq l_i) \tag{4.1}$$

where $\hat{l}_i$ is the predicted label of the image, l_i is the true label of the image, and $\mathcal{I}$ denotes the indicator function. A good classifier is one for which the test error rate is small [ISLR, Section 2.2].

An empirical comparison of the SRC with the more classical methods NN (ie: KNN with $K = 1$), NS (KNS with $K = 1$) and linear-kernel SVM is found in [Wri+09]. In this comparison, the researchers used the *Extended Yale B Database* consisting of $2\,414$ images of 38 individuals, as well as the *AR database* consisting of over $4\,000$ images of 126 individuals. For each subject, they used half of the images for training, and half of the images for testing. Then they paired up the different classifiers with different feature extraction and feature dimensions, and compared their test error rate[1].

We have included some of the results from [Wri+09] in Table 1. We have concentrated on the Laplacian faces scheme for feature extraction, as this seemed to perform well with all the different classifiers. As one can see from the table, the SRC performs quite well.

[1] In [Wri+09] they actually talk about *recognition rate*, which is $1-$ test error rate. My guess would be that the authors are the *"glass is half-full"*-kind of people.

	Feature dimension			
Classifier	30	56	120	504
SRC	0.125	0.083	0.060	0.035
NN	0.229	0.165	0.128	0.093
NS	0.110	0.096	0.081	0.066
SVM	0.280	0.150	0.060	0.023

Table 1: Test error rates reported in [Wri+09] for the SRC, NN, NS and SVM using the Laplacian faces scheme for feature extraction, and using the extended Yale B database

An even more impressive result appears when the different classifiers is applied to occluded or corrupted images. For images where parts of the face are occluded, the researchers found that the test error rate was below 0.02, even when up to 30% of the face where occluded. At 40% occlusion the test error rate increased to 0.1, and for 50% they reported a test error rate of 0.35. Meanwhile, the researchers reported a test error rate of > 0.2 at 30% occlusion for the nearest-neighbor methods, and at 50% they reported a test error rate between 0.5 and 0.7. Similar results were found for corrupted images.

4.2 Performance Issues

So far, we have only discussed the advantages of the SRC-based system for face recognition compared to the classical approaches. However, this gives an incomplete view of the system. In this section we will look at one major disadvantage to the SRC-based method, namely running time and memory use.

Runtime Analysis

We will begin by looking at how the runtime of Algorithm 3.1 will increase as the size of the problem increases (ie, as C, N, m and n increases). To do this we will use the Big-O notation.

Definition 4.1. Let f, g be two functions $f, g \colon \mathbb{N} \to \mathbb{R}$. Then $f(n) \in \mathcal{O}(g(n))$ if there exists a $c \in \mathbb{R}$ and a $N \in \mathbb{N}$ such that $f(n) \leq cg(n)$ for all $n > N$.

We can think of Big-O as an upper bound on the true runtime.

Step 1 and 3 of Algorithm 3.1 are obviously linear-time operations. More precisely, they have a runtime of $\mathcal{O}(N)$ and $\mathcal{O}(C)$ respectively. The bottleneck of the runtime is thus step 2. In Section 2.1 we saw how (P_1) could be recast as a LP problem. However, a famous result due to Klee and Minty renders this more or less useless. They showed that Simplex method using the largest-coefficient pivoting rule uses $\mathcal{O}(2^n)$ pivots worst case, where n is the number of decision variables [Van14, Section 4.4]. For our case this means that using the simplex method to implement Algorithm 3.1 will have a worst case runtime of $\mathcal{O}(2^N)$.

Various interior point methods exists, such as the path-following method. This algorithm uses newtons method to iteratively find better and better approximations to the optimal solution. However, even though the number of iterations no longer has exponential growth, since each iteration typically uses $\mathcal{O}(N^2)$ operations, this too is to slow for our case [EK12, Section 12.6]. This is because in a real world example, we will a very large number of training images N.

The last algorithm we will look at is the orthogonal matching pursuit used to create Figure 2.2. This algorithms solves a series of least-squares approximations. This too is an approach that scales badly, which is why it's mostly used for small problems [FR13, Secion 3.2].

It is worth noting that some optimization algorithms with better per-iteration runtime exists, such as the Augmented Lagrange multiplier method described in [FR13, Alg. 12.2]. Other algorithms, such as homotopy algorithms, make use of the fact that the solution is sparse, and are able to recover the ℓ_1-minimum of an s-sparse vector in $\mathbb{R}^N$ in $\mathcal{O}(s^3 + N)$ time [Wri+09].

Memory Usage

As we now have seen, the issue with runtime can be somewhat dealt with. Still, another issue remains, and that is memory usage.

size of data

gange med matrise, vs å allokere hele mat.

5 Conclusions

results, consequences, etc

References

[Ant16] Vegard Antun. *Master thesis code*. 2016. URL: https://bitbucket.org/vegarant/code-thesis (visited on 03/23/2017).

[BL13] Kurt Bryan and Tanya Leise. "Making Do with Less: An Introduction to Compressed Sensing." In: *Siam Review* 55.3 (2013), pp. 547–566.

[BW14] Farooq Ahmad Bhat and M. Arif Wani. "Performance Comparison of Major Classical Face Recognition Techniques." In: *IEEE 13th International Conference on Machine Learning and Applications (ICMLA)* (2014), pp. 521–528.

[EK12] Yonina C. Eldar and Gitta Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge university press, 2012. Chap. 12.

[FR13] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhauser, 2013.

[ISLR] Gareth James et al. *An Introduction to Statistical Learning with Applications in R*. Springer, 2013.

[LHK05] Kuang-Chih Lee, Jeffrey Ho, and David J. Kriegman. "Acquiring linear subspaces for face recognition under variable lighting." In: *IEEE Transactions on pattern analysis and machine intelligence* 27.5 (2005), pp. 684–698.

[Rya16] Øyvind Ryan. *Linear algebra, signal processing and wavelets. A unified approach*. 2016. URL: https://github.com/oyvindry/applinalgcode.

[Van14] Robert J. Vanderbei. *Linear Programming*. 4th ed. Springer, 2014.

[WM10] John Wright and Yi Ma. "Dense Error Correction Via ℓ^1-Minimization." In: *IEEE Transactions on Information Theory* 56.7 (July 2010), pp. 3540–3560.

[Wri+09] John Wright et al. "Robust face recognition via sparse representation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.2 (Feb. 2009), pp. 210–227.