UiO : **Department of Mathematics**
University of Oslo

Title

Optional Subtitle

Andras Filip Plaian

Master's Thesis, Autumn 2021

Abstract

Add new section about results in

Acknowledgements

Rewrite this.

Contents

List of Figures

CHAPTER 1

Introduction

Write introduction here.

Outline

The rest of the text is organised as follows:

CHAPTER 2

Compressed Sensing

In an underdetermined system of linear equations there are fewer equations than unknowns. In mathematical terms this can be stated as the matrix equation $A\mathbf{x} = \mathbf{y}$ where $A \in \mathbb{C}^{m \times N}$, $\mathbf{x} \in \mathbb{C}^N$, $\mathbf{y} \in \mathbb{C}^m$ and $m < N$. From classical linear algebra it follows that this equation is unsolvable in the general case, however, given certain conditions, it is possible to find an exact or an estimate of a solution.

The research area associated with these assumptions is called Compressed Sensing (CS). The goal of this chapter is to introduce the reader to the needed concepts to be able to determine the needed conditions for being able to find an efficient, stable and robust reconstruction method of an estimate or solution of the *inverse problem* $\mathbf{x} = A^{-1}\mathbf{y}$ by exploiting the theory of CS.
We also want to be able to have some answer to the following questions:

- What properties and conditions does A need to have in order for $\mathbf{x}$ to be reconstructable?
- Does a suitable algorithm for finding $\mathbf{x}$ exist?

2.1 Main Assumptions: Sparsity and Compressibility

The main assumption we do about the vector $\mathbf{x}$ we are trying to recover is that it is sparse. Informally, a vector is *sparse* if most of its components is zero, particularly if we find an appropriate basis for the given vector $\mathbf{y} = A\mathbf{x}$. For instance, natural images are sparse in the wavelet domain. Before we define sparsity formally, we recall the definition of the *support* of a given vector.

Definition 2.1. The support of a vector $\mathbf{x} \in \mathbb{C}^N$ is the index set of its non-zero entries, that is:

$$supp(\mathbf{x}) := \{i \in [N] : x_i \neq 0\}.$$

In Compressive Sensing it is customary to abuse the l_0 notation to denote the cardinality of the support set , i.e. the number of non-zero entries of a vector. The $\|\cdot\|_0$ fails to be a norm since it does not satisfy the scaling property of norms. We can now define *s-sparse* vectors.

Definition 2.2. The vector $\mathbf{x} \in \mathbb{C}^N$ is called *s-sparse* if it has no more than s non-zero entries, that is if: $\|\mathbf{x}\|_0 \leq s$

The notion of sparsity is an ideal one, meaning that in the real world it may very well be that the vector we are trying to recover is only close to being sparse. In order for CS to handle more problems, we may also be interested in the following notion of *compressibility*.

Definition 2.3. For $p > 0$, the measure of a vectors compressibility is given by the l_p-error of best s-term approximation to $\mathbf{x} \in \mathbb{C}^N$ defined by

$$\sigma(\mathbf{x})_p := inf\{\|\mathbf{x} - \mathbf{z}\|_p, \mathbf{z} \in \mathbb{C}^N \text{ is } s\text{-sparse}\}$$

Informally, a vector is compressible if the l_p-error decays quickly in s. For instance in image processing, if we can describe most of the variance in image data with a few components of a vector.

2.2 Basic Requirements of A

What properties does A need to have in order to have the possibility to reconstruct every s-sparse vector *regardless* of the reconstruction method? For instance, what is the minimal number of rows that A needs to have such that it would be possible to find every s-sparse vector? We have the following result:

Theorem 2.4. *For a given matrix $A \in \mathbb{C}^{m \times N}$, the following statements are equivalent:*

a) *Every s-sparse vector $\mathbf{x} \in \mathbb{C}^N$ is the unique s-sparse solution of $A\mathbf{z} = A\mathbf{x}$, that is, if $A\mathbf{x} = A\mathbf{z}$ and both $\mathbf{z}$ and $\mathbf{x}$ are s-sparse, then $\mathbf{z} = \mathbf{x}$.*

b) *The null space $ker(A)$ does not contain any $2s$-sparse vector other than the zero vector, that is, $ker(A) \cap \{\mathbf{z} \in \mathbb{C}^N : \|\mathbf{z}\|_0 \leq 2s\} = \{\mathbf{0}\}$.*

c) *For every $S \subset [N]$ with $card(S) \leq 2s$, the submatrix A_S is injective.*

d) *Every set of $2s$ columns of A is linearly independent.*

Proof. b) $\Longrightarrow$ a) Let $\mathbf{x}, \mathbf{z}$ be s-sparse vectors, not necessarily unique, such that $A\mathbf{x} = A\mathbf{z}$. Then $\mathbf{x} - \mathbf{z}$ is at most $2s$-sparse and using linearity of A we get $A(\mathbf{x} - \mathbf{z}) = \mathbf{0}$. If b) holds, then the kernel only contains the zero vector of $2s$-sparse vectors, then the only way $A(\mathbf{x} - \mathbf{z}) = \mathbf{0}$ is true, is if $\mathbf{x} = \mathbf{z}$, thus $\mathbf{x}$ is the unique s-sparse vector.
a) $\Longrightarrow$ b) Assume that $\mathbf{x}$ is the unique s-sparse vector such that $A\mathbf{x} = A\mathbf{z}$. Let $\mathbf{v} \in ker(A)$ be $2s$-sparse. Since $\mathbf{v}$ is $2s$-sparse, there exist a way to construct $\mathbf{v}$ from $\mathbf{x} - \mathbf{z}$ where $\mathbf{x}, \mathbf{z}$ are both s-sparse and such that they have no components

in common. Assuming a) holds, then we have $\mathbf{x} = \mathbf{z} = \mathbf{v} = \mathbf{0}$.
b) $\iff$ c) $\iff$ d) For a $2s$-sparse vector $\mathbf{v}$ with $S = \operatorname{supp} \mathbf{v}$, if $\mathbf{a}_1, ..., \mathbf{a}_S$ are the column vectors of the submatrix A_S made from A, we get $A\mathbf{v} = A_S\mathbf{v_S}$. Noting that $S = \operatorname{supp} \mathbf{v}$ ranges through all possible subsets of [N] with card$(S) \leq 2s$ ranging through all the possible $2s$-sparse $\mathbf{v}$ vectors. Then in this view, linear independence d), injectivity c) and that the kernel is trivial b) are equivalent statements about A.

■

With a little knowledge about how to construct invertible matrices, we can derive the following result from the above theorem.

Corollary 2.5. *For any integer $N \geq 2s$, there exists a matrix $A \in \mathbb{C}^{m\times N}$ with $m = 2s$ rows such that every s-sparse vector $\mathbf{x} \in \mathbb{C}^N$ can be recovered from the vector $\mathbf{y} = A\mathbf{x} \in \mathbb{C}^m$.*

Proof. Let $t_1, t_2, ..., t_N$ be strictly positive numbers and consider the matrix $A \in \mathbb{C}^{m\times N}$ with $m = 2s$ defined by

$$A = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ t_1 & t_2 & \cdots & t_N \\ \vdots & \vdots & \cdots & \vdots \\ t_1^{2s-1} & t_2^{2s-1} & \cdots & t_N^{2s-1} \end{bmatrix}$$

Now define a square submatrix $A_S \in \mathbb{C}^{m\times m}$ indexed by $S = \{i_1 < ... < i_m\}$ as

$$A_S = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ t_{i_1} & t_{i_2} & \cdots & t_{i_m} \\ \vdots & \vdots & \ddots & \vdots \\ t_{i_1}^{2s-1} & t_{i_2}^{2s-1} & \cdots & t_{i_m}^{2s-1} \end{bmatrix}$$

The given submatrix A_S can be recognized as the transposed *Vandermonde matrix*. A result about Vandermonde matrices is that its determinant, in view of A_S, can be expressed as $\det(A_S) = \prod k < l \leq m(t_{i_l} - t_{i_k})$. Since the t's were defined to be strictly positive it follows that $\det(A_S) > 0$, and hence A_S is invertible. The Invertible Matrix Theorem then tells us that A_S is injective which means that condition *iii*) in Theorem 2.4 holds and then we have a unique s-sparse vector $\mathbf{x} \in \mathbb{C}^N$ such that $A\mathbf{z} = A\mathbf{x}$. ■

2.3 Finding Suitable Reconstruction Algorithms

Recall that the compressed sensing problem is to find an s-sparse vector $\mathbf{x} \in \mathbb{C}^N$ given $\mathbf{y} = A\mathbf{x}$. Abusing the customary l_0 notation for the number of non-zero entries, we can reformulate this problem as an optimization problem, that is:

$$\text{minimize } \|\mathbf{z}\|_0 \quad \text{subject to } A\mathbf{z} = \mathbf{y}. \qquad (P_0)$$

A naive approach to solving this could be to look at it as a combinatorial optimization problem and use brute force to calculate every square system $A_S^*A_S\mathbf{x} = A_S^*\mathbf{y}$, for $\mathbf{x} \in \mathbb{C}^S$ where S runs through all possible subsets of

$[N]$ with size s. This might very well work on small sizes of N, but the total amount of subsets the algorithm has to go through is determined by the formula $\binom{N}{s}$. A quick illustration shows that with $N = 1000, s = 10$ we have $\binom{1000}{10} \geq \binom{1000}{10}^{10} = 10^{20}$ linear systems of size 10×10 to solve. Assuming one could solve each system in 10^{-10} seconds, it would still take 10^10 seconds, ie. several human lifespans, thus the brute force approach is completely unpractical for sufficiently large N.

P_0 is NP-hard

It's not only the brute force approach that is not bounded by a polynomial expression, it might be that there doesn't exist an approach bounded by a polynomial expression at all. In fact, we have the following result:

Theorem 2.6. *The l_0-minimization problem for an arbitrary matrix $A \in \mathbb{C}^{m \times N}$ and a vector $\mathbf{y} \in \mathbb{C}^m$ is NP-hard.*

Before proving the result, let us recall some of the terminology:

- The class of P problems consists of all decision problems for which there exists a polynomial-time algorithm, ie. an input size bounded by a polynomial expression,for finding a solution.
- The class of NP problems, not to be confused with NP-hard, consists of all decision problems for which there exists a polynomial-time algorithm *certifying* a solution.
- The class of NP-hard problems consists of all problems for which *solving* algorithm could be transformed in polynomial time into a *solving* algorithm for any NP-problem.

This means that if we can show that a given problem solves an other problem belonging a certain class of problems, then the given problems also belongs to that same class.

Proof. If we can show that a known NP-hard problem can be reduced in polynomial time to the l_0-minimization problem then l_0-minimization itself is NP-hard.
Let The Exact Cover by 3-sets problem be our known NP-hard problem. The Exact Cover by 3-sets problem says that given a collection $\{\mathcal{C}_i, i \in [N]\}$ of 3-element subsets of $[m]$, does there exist an exact partition or cover of the set $\{1, 2, ..., m\} = [m]$?

Let $\{\mathcal{C}_i, i \in [N]\}$ be the collection of 3-element subsets of $[m]$. Define vectors $\mathbf{a}_1, ..., \mathbf{a}_N \in \mathbb{C}^m$ by

$$a_{ij} = \begin{cases} 1 \text{ if } j \in \mathcal{C}_i, \\ 0 \text{ if } j \notin \mathcal{C}_i. \end{cases}$$

where $j \in [m]$.

Define a matrix $A \in \mathbb{C}^{m \times N}$ and a vector $\mathbf{y} \in \mathbb{C}^m$ by

$$A = \begin{bmatrix}\mathbf{a}_1 \cdots \mathbf{a}_N\end{bmatrix} \text{ and } \mathbf{y} = [1, ..., 1]^T.$$

Since $N \le \binom{m}{3}$, this construction can be done in polynomial time since $\binom{m}{3} = \frac{m(m-1)(m-2)}{3!}$ which is a 3rd degree polynomial in m.

If a vector $\mathbf{z} \in \mathbb{C}^N$ satisfies $A\mathbf{z} = \mathbf{y}$, then all the m components of $A\mathbf{z}$ are nonzero and $\|A\mathbf{z}\|_0 = m$. Since each vector $\mathbf{a}_i$ has exactly 3 nonzero components, the vector $A\mathbf{z} = \sum_{j=1}^N z_j \mathbf{a}_j$ has at most 3 $\|z\|_0$ nonzero components, $\|A\mathbf{z}\|_0 \le 3\|z\|_0$ and consequently $\|z\|_0 \ge m/3$. Now consider the l_0-minimization and let $\mathbf{x} \in \mathbb{C}^N$ be the output. We separate two cases:

1. If $\|\mathbf{x}\|_0 = m/3$, then the collection $\{\mathcal{C}_j, j \in \text{supp}(\mathbf{x})\}$ forms an exact cover of $[m]$, for otherwise the m components of $A\mathbf{x} = \sum_{j=1}^N x_j \mathbf{a}_j$ would not all be nonzero.

2. If $\|\mathbf{x}\|_0 > m/3$, then no exact cover $\{\mathcal{C}_j, j \in J\}$ can exist, for otherwise the vector $\mathbf{z} \in \mathbb{C}^N$, defined by $z_j = 1$ if $j \in J$ and $z_j = 0$ if $j \neq J$, would satisfy $A\mathbf{z} = \mathbf{y}$ and $\|\mathbf{z}\|_0 = m/3$, contradicting the l_0-minimality of $\mathbf{x}$.

This shows that solving l_0-minimization problem enables one to solve the Exact Cover by 3-sets problem and consequently l_0-minimization problem is NP-hard. ■

P_q where $0 < q < 1$ also NP-hard

If P_0 optimization is intractable, would P_q optimization be a better approach? Not according to the following proposition:

Proposition 2.7. *l_q-minimization for $0 < q < 1$ is NP-hard.*

Proof. The proof goes along the same lines as proving the NP-hardness of l_0-minimization, that is if l_q-minimization can help to verify that a given NP-hard problem has a solution, that is if the minimum of $\|\mathbf{w}\|_q$ subject to $A\mathbf{w} = \mathbf{y}$, equals n, then l_q-minimization must necessarily also belong to the same class of NP-hard or NP-complete problems.
The *partition problem* consists, given integers $a_1, a_2, ..., a_n$, in deciding whether there exists two sets $I, J \subset [n]$ such that $I \cap J = \emptyset$, $I \cup J = [n]$ and $\sum a_i = \sum a_j$ for $i, j \in I, J$ respectively.
Define

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n & -a_1 & -a_2 & \cdots & -a_n \\ 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & 0 & \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 1 \end{bmatrix} \text{ and } \mathbf{y} = [0, 1, 1, ..., 1]^T.$$

Let $\mathbf{x}$ and $\mathbf{z}$ be the first and second half of the input vector $\mathbf{w}$ to A respectively. That $A\mathbf{w} = \mathbf{y}$ implies that $\sum_{i=1}^{n} a_i x_i = \sum_{i=1}^{n} a_i z_i$ and $x_i + z_i = 1$ for all i. We seek to minimize $\sum_{i=1}^{n}(x_i^p + z_i^p)$ under these constraints.
The minimum value for the objective function when only the constraints $x_i + y_i$ are considered is n, and this occurs if and only if only 0's and 1's are involved. If the minimum $\|\mathbf{w}\|_q$ subject to $A\mathbf{w} = \mathbf{y}$ is n, then it will also be n when only the $x_i + y_i = 1$ constraints are considered. This means that either x_i or z_i is equal to 1, for all i. Let I be the set of i's where x_i=1, and J be the set of i's where z_i=1. We then have $I \cap J = \emptyset$ and $I \cup J = [n]$, ie. a partition on the form we seek.
Defining x_i as 1 when $i \in I$ and defining $\mathbf{z}$ similarly, we obtain a vector $\mathbf{w}$ where all the constraints are fulfilled and where the minimum n is obtained.

■

P_q where $q > 1$

If both l_0 and l_q-minimization for $0 < q < 1$ is NP-hard, how about l_q-minimization for $q > 1$? Unfortunately, this optimization problem fails to recover even 1-sparse vectors. We have the following proposition:

Proposition 2.8. *Let $q > 1$ and let A be a $m \times N$ matrix with $m < N$. Then there exists a 1-sparse vector which is not a minimizer of P_q.*

Proof. Since $m < N$, we have to from linear algebra that the kernel of A is non-trivial. Hence there must exists a $\mathbf{v} \neq \mathbf{0} \in \ker(A)$ such that $A\mathbf{v} = \mathbf{0}$. Assume, for the sake of contradiction, that all standard basis vectors $\mathbf{e}_j$ are minimizers. Choose a j so that $v_j \neq 0$. Then

$$\|\mathbf{e}_j + t\mathbf{v}\|_q^q = |1 + tv_j|^q + \sum_{k\neq j}|tv_k|^q = |1 + tv_j|^q + |t|^q \sum_{k\neq j}|v_k|^q$$

Define two functions such that

$$g_+(t) = (1 + tv_j)^q + t^q \sum_{k\neq j}|v_k|^q$$

$$g_-(t) = (1 + tv_j)^q + (-t)^q \sum_{k\neq j}|v_k|^q$$

For $|t| < 1/v_j$, $\|\mathbf{e}_j + t\mathbf{v}\|_q^q$ coincides with g_+ for $t \geq 0$, and g_- for $t \leq 0$. Taking the derivatives of g_+ and g_- we get

$$g'_+(t) = qv_j(1 + tv_j)^{q-1} + qt^{q-1} \sum_{k\neq j}|v_k|^q$$

$$g'_-(t) = qv_j(1 + tv_j)^{q-1} - q(-t)^{q-1} \sum_{k\neq j}|v_k|^q$$

Taking the limit of the two derivatives and then for $q > 1$ we get

$$\lim_{t\to 0+} g'_+(t) = \lim_{t\to 0-} = g'_-(t) = qv_j$$

This means that near 0, $\|\mathbf{e}_j + t\mathbf{v}\|_q^q$ has derivative arbitrarily near $qv_j \neq 0$. Since a linear function with derivative qv_j has no minimum near 0, then with $t = 0$, $\mathbf{e}_j$ can't be a minimizer of P_q, which consequently contradicts our assumption that all standard basis vectors $\mathbf{e}_j$ are minimizers of P_q.

■

The special case P_q where $q = 1$

Recall that in an optimization problem we try to minimize or maximize an objective function subject to constraint functions. If all the constraint functions are also convex functions, then the optimization problem becomes a convex optimization problem. Thus both l_0 and l_q-minimization for $0 < q < 1$ are non-convex optimization problems. If all the constraint functions are linear functions, then the following convex optimization problem

$$\text{minimize } \|\mathbf{z}\|_1 \quad \text{subject to } A\mathbf{z} = \mathbf{y}. \tag{P_1}$$

is also a linear program. Efficient algorithms exists for solving LP's. For instance, the Simplex method, behaves like a polynomial-time algorithm for solving real-life LP problems. l_1 -minimization is also known *basis pursuit* and during the next sections we shall see that given certain assumptions on A we can actually solve P_1 and that the recovered solutions will be sparse.

2.4 Basis Pursuit

In order to show how BP can identify the solution to (P_0), we need to introduce some definitions and theorems, an important property which our matrix must have, is the so-called Null Space Property of a matrix A:

Definition 2.9. A matrix $A \in \mathbb{C}^{m\times N}$ is said to satisfy the *Null Space Property (NSP)* relative to a set $S \subset \{1, 2, ..., N\}$ if

$$\min \|\mathbf{v}_S\|_1 < \|\mathbf{v}_{\overline{S}}\|_1 \text{ for all } \mathbf{v} \in \ker A \setminus \{\mathbf{0}\}$$

A is said to satisfy the *Null Space Property of order s* if it satisfies the null space property relative to any set S $\subset \{1, 2, ..., N\}$ with $|\mathrm{S}| \leq s$.

The following theorem shows that the NSP of a matrix is a sufficient condition in order to solve (P_0).

Theorem 2.10. *Given a matrix $A \in \mathbb{C}^{m\times N}$, every vector $\mathbf{x} \in \mathbb{C}^N$ supported on a set S is the unique solution (P_1) with $A\mathbf{x} = \mathbf{y}$ if and only if A satisfies the NSP relative to S.*

Furthermore, if the set S varies, then every s-sparse vector $\mathbf{x} \in \mathbb{C}^N$ is the unique solution to (P_1) with $A\mathbf{x} = \mathbf{y}$ if and only if A satisfies the NSP of order s.

Proof. Let S be a fixed index set, and assume that every vector $\mathbf{x} \in \mathbb{C}^N$ supported on this set, is the unique minimizer of (P_1). From the assumption it follows that for $\mathbf{v} \in \ker A \setminus \{\mathbf{0}\}$, the vector $\mathbf{v}_S$ is the unique minimizer of (P_1). Since $A(\mathbf{v}_S + \mathbf{v}_{\overline{S}}) = \mathbf{0}$ and $-\mathbf{v}_S \neq \mathbf{v}_{\overline{S}}$, from the minimality assumption we must have that $\|-\mathbf{v}_S\|_1 < \|\mathbf{v}_{\overline{S}}\|_1$. This established the NSP relative to S.

Conversely, assume that NSP relative to S holds. Let $\mathbf{x} \in \mathbb{C}^N$ be supported on S and a vector $\mathbf{z} \in \mathbb{C}^N$, $\mathbf{z} \neq \mathbf{x}$, such that $A\mathbf{z} = A\mathbf{x}$. Following the rules for norms and taking complements for the support of a set , we obtain

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

Which shows that $\mathbf{x}$ obtains the unique minimum.

To prove the second part of the theorem, let S vary and assume that every s-sparse vector $\mathbf{x}$ is found by solving (P_1) subject to $A\mathbf{x} = \mathbf{y}$. Let $\mathbf{z}$ be the solution to P_0 subject to $A\mathbf{x} = \mathbf{y}$ then $\|\mathbf{z}\|_0 \leq \|\mathbf{x}\|_0$ so that also $\mathbf{z}$ is s-sparse. But since every s-sparse vector is the unique minimizer of (P_1), we have that $\mathbf{x} = \mathbf{z}$ and the result follows. ■

Minimum Number Of Rows For Basis Pursuit

From the results above it is clear that if a matrix possesses the NSP of order s, the BP will solve (P_1). Next we will introduce a theorem that can identify when A has the NSP of order s.

Theorem 2.11. *Given a matrix $A \in \mathbb{C}^{m\times N}$, then every set of $2s$ columns of A is linearly independent if and only if A satisfies the NSP of order s.*

Proof. Assume that every $2s$ columns of A is linearly independent, then from The Invertible Matrix Theorem, we have that the kernel of A does not contain any other $2s$-sparse vector other than $\mathbf{0}$. Now let $\mathbf{x}$, and $\mathbf{z}$ be s-sparse with $A\mathbf{z} = A\mathbf{x}$. Then $A(\mathbf{x} - \mathbf{z}) = \mathbf{0}$ and $\mathbf{x} - \mathbf{z}$ is $2s$-sparse, but since $\ker A \setminus \{\mathbf{0}\}$ is empty, we must have $\mathbf{x} = \mathbf{z}$, but this implies that the NSP of order s holds.
Conversely, assume that the kernel of A does not contain any other $2s$-sparse vector other than $\mathbf{0}$, then for any set S with $\operatorname{card}(S) = 2s$, $A_S\mathbf{x} = \mathbf{0}$ only has the trivial solution and thereby $2s$ linearly independent columns. ■

From these results we can derive that :

Corollary 2.12. *In order for Basis Pursuit to solve (P_0), the matrix $A \in \mathbb{C}^{m\times N}$ has to satisfy:*

$$m \geq 2s$$

Proof. Assume that it is possible to uniquely recover any s-sparse vector $\mathbf{x}$ from $\mathbf{y} = A\mathbf{x}$. Then, by Theorem 2.4, statement a) holds, and consequently so does d), that is, that every set of $2s$ columns of A is linearly independent. Thus $\operatorname{rank}(A) \geq 2s$, but we also have that the rank of a matrix can not be greater than the number of rows m, so we must have rank$(A) \leq m$. Combining and arranging these two inequalities, we get that

$$2s \leq \text{rank}(A) \leq m$$

which implies, $m \geq 2s$, the inequality in the corollary. ■

Stability

Basis Pursuit features another important property, namely *stability*. This property tackles sparsity defects, that is, to recover a vector $\mathbf{x} \in \mathbb{C}^N$ whose error to the true underlying vector is controlled by its distance to an s-sparse vector.

Definition 2.13. A matrix $A \in \mathbb{C}^{m\times N}$ is said to satisfy the *stable null space property* with $0 < \rho < 1$ relative to a set $S \subset [N]$ if

$$\|\mathbf{v}_S\|_1 \leq \rho \|\mathbf{v}_{\overline{S}}\|_1 \text{ for all } \mathbf{v} \in \ker A$$

Furthermore, it satisfies the stable null space of order s if it satisfies the stable null space property relative to any set $S \subset [N]$ with card$(S) \leq s$.

If a matrix satisfies the stable null space property of order s, then the l_1-error to the true underlying vector $\mathbf{x}$ is given by

$$\|\mathbf{x} - \mathbf{x}'\|_1 \leq \frac{2(1+\rho)}{1-\rho}\sigma_s(\mathbf{x})_1 \tag{2.1}$$

The next result will give us a condition for when, the inequality (2.1) and the stable null space property, will hold.

Theorem 2.14. *The matrix $A \in \mathbb{C}^{m\times N}$ satisfies the stable null space property with constant $0 < \rho < 1$ relative to S if and only if*

$$\|\mathbf{z} - \mathbf{x}\|_1 \leq \frac{1+\rho}{1-\rho}(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\|\mathbf{x}_{\overline{S}}\|_1) \tag{2.2}$$

for all vectors $\mathbf{x}, \mathbf{z}, \in \mathbb{C}^N$ with $A\mathbf{z} = A\mathbf{x}$.

Before proving Theorem 2.14, we show that (2.2) implies (2.1). Let S be the set of the s largest non-negative components of $\mathbf{x}$, then $\|\mathbf{x}_{\overline{S}}\|_1 = \sigma_s(\mathbf{x})_1$. If $\mathbf{x}'$ is a minimizer of (P_1), then $\|\mathbf{x}'\|_1 \leq \|\mathbf{x}\|_1$ and $A\mathbf{x}' = A\mathbf{x}$. The right-hand side of the inequality (2.2) with $\mathbf{z} = \mathbf{x}'$ will then be equal to the right hand side of inequality (2.1).

Proof. To prove Theorem 2.14, assume that the matrix A satisfies inequality (2.2) for all vectors $\mathbf{x}, \mathbf{z} \in \mathbb{C}^N$ with $A\mathbf{z} = A\mathbf{x}$. Given a vector $\mathbf{v} \in \ker(A)$, since $A\mathbf{v}_{\overline{S}} = A(-\mathbf{v}_S)$, we can apply (2.2) with $\mathbf{x} = -\mathbf{v}_S$ and $\mathbf{z} = \mathbf{v}_{\overline{S}}$. This gives us that

$$\|\mathbf{v}\|_1 \leq \frac{1+\rho}{1-\rho}(\|\mathbf{v}_S\|_1 - \|\mathbf{v}_{\overline{S}}\|_1)$$

Rearranging and cancelling equal terms, this can be written as

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

which we recognize as the stable NSP with constant $0 < \rho < 1$.

Conversely, assume that the matrx A satisfies the stable NSP with constant $0 < \rho < 1$ relative to S. Then for $\mathbf{x}, \mathbf{z} \in \mathbb{C}^N$ with $A\mathbf{z} = A\mathbf{x}$, we get that $\mathbf{v} = \mathbf{z} - \mathbf{x} \in \ker(A)$ yields

$$\|\mathbf{v}_S\|_1 \le \rho\|\mathbf{v}_{\overline{S}}\|_1. \tag{2.3}$$

Since $\|\mathbf{v}_{\overline{S}}\|_1 = \|(\mathbf{z} - \mathbf{x})_{\overline{S}}\|_1$, we can rewrite (2.3) into

$$\|\mathbf{v}_{\overline{S}}\|_1 \le \|\mathbf{z}_1\| - \|\mathbf{x}_1\| + \rho\|\mathbf{v}_{\overline{S}}\|_1 + 2\|\mathbf{x}_{\overline{S}}\|_1.$$

Since $\rho < 1$, this can be rewritten as

$$\|\mathbf{v}_{\overline{S}}\|_1 \le \frac{1}{1-\rho}(\|\mathbf{z}_1\| - \|\mathbf{x}_1\| + 2\|\mathbf{x}_{\overline{S}}\|_1).$$

Using (2.3) once again, we chain the inequalities to get,

$$\|\mathbf{v}\|_1 = \|\mathbf{v}_{\overline{S}}\|_1 + \|\mathbf{v}_S\|_1 \le (1+\rho)\|\mathbf{v}_{\overline{S}}\|_1 \le \frac{1+\rho}{1-\rho}(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\|\mathbf{x}_{\overline{S}}\|_1)$$

which is (2.2), the desired inequality.

■

Robustness

Basis Pursuit features another important property, namely *robustness.* This property tackles noise.

Definition 2.15. Given $q \ge 1$, the matrix $A \in \mathbb{C}^{m\times N}$ is said to satisfy the l_q*-robust null space property* (rNSP), with constants $0 < \rho < 1$ and $\tau > 0$, relative to a set $S \subset [N]$ if

$$\|\mathbf{v}_S\|_q < \frac{\rho}{s^{1-1/q}}\|\mathbf{v}_{\overline{S}}\|_1 + \tau\|A\mathbf{v}\| \text{ for all } \mathbf{v} \in \mathbb{C}^N.$$

Furthermore, it satisfies the l_q-rNSP of order s if it satisfies the l_q-rNSP relative to any set $S \subset [N]$ with $\text{card}(S) \le s$.

In order to introduce a more general result using l_q-rNSP, we need an intermediate result that holds for the ordinary rNSP, that is setting $q = 1$.

Theorem 2.16. *The matrix $A \in \mathbb{C}^{m\times N}$ satisfies the robust null space property with constants $0 < \rho < 1$ and $\tau > 0$ relative to S if and only if*

$$\|\mathbf{z}-\mathbf{x}\|_1 \leq \frac{1+\rho}{1-\rho}(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\|\mathbf{x}_{\overline{S}}\|_1) + \frac{2\tau}{1-\rho}\|A(\mathbf{z}-\mathbf{x})\| \tag{2.4}$$

for all vectors $\mathbf{x}, \mathbf{z} \in \mathbb{C}^N$.

Proof. Very similar to the proof of Theorem 2.14, will therefore use results from there when appropriate.
Assume A satisfies (2.4) for all vectors $\mathbf{x}, \mathbf{z} \in \mathbb{C}^N$. Let $\mathbf{v} \in \mathbb{C}^N$ be so that $\mathbf{v} = \mathbf{z} - \mathbf{x}$ with $\mathbf{z} = \mathbf{v}_{\overline{S}}$ and $\mathbf{x} = -\mathbf{v}_S$ which gives

$$\|\mathbf{v}\|_1 \leq \frac{1+\rho}{1-\rho}(\|\mathbf{v}_{\overline{S}}\|_1 - \|\mathbf{v}_S\|_1) + \frac{2\tau}{1-\rho}\|A\mathbf{v}\|.$$

Multiplying by $(1-\rho)$ on both sides and expanding $\|\mathbf{v}\|_1$ on the left side gives

$$(1-\rho)(\|\mathbf{v}_S\|_1 + \|\mathbf{v}_{\overline{S}}\|_1) \leq (1+\rho)(\|\mathbf{v}_{\overline{S}}\|_1 - \|\mathbf{v}_S\|_1) + 2\tau\|A\mathbf{v}\|.$$

Expanding, cancelling equal terms and dividing by 2 on both sides results in

$$\|\mathbf{v}_S\|_1 \leq \rho\|\mathbf{v}_{\overline{S}}\|_1 + \tau\|A\mathbf{v}\|$$

which is the rNSP with constants $0 < \rho < 1$ and $\tau > 0$.

For the converse, assume A satisfies rNSP with constants $0 < \rho < 1$ and $\tau > 0$ relative to S. For $\mathbf{z}, \mathbf{x} \in \mathbb{C}^N$ with $\mathbf{v} = \mathbf{z} - \mathbf{x}$, the rNSP gives

$$\|\mathbf{v}_S\|_1 \leq \rho\|\mathbf{v}_{\overline{S}}\|_1 + \tau\|A\mathbf{v}\|,$$

$$\|\mathbf{v}_{\overline{S}}\|_1 \leq \|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + \|\mathbf{v}_S\|_1 + 2\|\mathbf{x}_{\overline{S}}\|_1$$

Combining these two inequalities and using the rNSP again we arrive at

$$\|\mathbf{v}\|_1 = \|\mathbf{v}_S\|_1 + \|\mathbf{v}_{\overline{S}}\|_1 \leq (1+\rho)\|\mathbf{v}_S\|_1 + \tau\|A\mathbf{v}\|$$
$$\leq \frac{1+\rho}{1-\rho}(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\|\mathbf{x}_{\overline{S}}\|_1) + \frac{2\tau}{1-\rho}\|A\mathbf{v}\|,$$

which is the desired inequality.

■

Theorem 2.17. *Given $1 \leq p \leq q$, suppose that the matrix $A \in \mathbb{C}^{m\times N}$ satisfies the l_q-robust null space property of order s with constants $0 < \rho < 1$ and $\tau > 0$. Then, for any $\mathbf{x}, \mathbf{z} \in \mathbb{C}^N$,*

$$\|\mathbf{x}-\mathbf{z}\|_p \leq \frac{C}{s^{1-1/p}}(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\sigma_s(\mathbf{x})_1) + Ds^{1/p-1/q}\|A(\mathbf{z}-\mathbf{x})\|,$$

where $C := (1+\rho)^2/(1-\rho)$ and $D := (3+\rho)\tau/(1-\rho)$.

In order to prove the above result, we need the following little lemma:

Lemma 2.18. *For any $q > p > 0$ and any $\mathbf{x} \in \mathbb{C}^N$, the inequality*

$$\sigma_s(\mathbf{x})_q \leq \frac{c_{p,q}}{s^{1/p-1/q}} \|\mathbf{x}\|_p$$

holds with

$$c_{p,q} = \Big[\Big(\frac{p}{q}\Big)^{p/q}\Big(1-\frac{p}{q}\Big)^{1-p/q}\Big]^{1/p} \leq 1.$$

Particularly for our proof, with the choice $p = 1$ and $q = p$ gives

$$\sigma_s(\mathbf{x})_p \leq \frac{1}{s^{1-1/p}} \|\mathbf{x}\|_1$$

Proof. Proof is omitted here for sake of readability, but can be found in the appendix. ■

Proof. (Theorem 2.17)
Remark that the l_q-rNSP implies the l_1-rNSP and the l_p-rNSP ($p \leq q$) in the forms

$$\|\mathbf{v}_S\|_1 \leq \rho\|\mathbf{v}_{\overline{S}}\|_1 + \tau s^{1-1/q}\|A\mathbf{v}\|, \tag{2.5}$$

$$\|\mathbf{v}_S\|_p \leq \frac{\rho}{s^{1-1/q}}\|\mathbf{v}_{\overline{S}}\|_1 + \tau s^{1/p-1/q}\|A\mathbf{v}\|, \tag{2.6}$$

for all $\mathbf{v} \in \mathbb{C}^N$ and all $S \subset [N]$ with card(S)$\leq s$. In view of inequality (2.5), applying Theorem 2.16 with S chosen as an index set of s largest non-negative components of $\mathbf{x}$ we get

$$\|\mathbf{z}-\mathbf{x}\|_1 \leq \frac{1+\rho}{1-\rho}(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\sigma_s(\mathbf{x})_1) + \frac{2\tau}{1-\rho}s^{1-1/q}\|A(\mathbf{z}-\mathbf{x})\|. \tag{2.7}$$

Next, we choose S as an index set of s largest non-negative components of $\mathbf{z}-\mathbf{x}$, and apply Lemma 2.18 to it, which yields

$$\|\mathbf{z}-\mathbf{x}\|_p \leq \|(\mathbf{z}-\mathbf{x})_{\overline{S}}\|_p + \|(\mathbf{z}-\mathbf{x})_S\|_p \leq \frac{1}{s^{1-1/p}}\|\mathbf{z}-\mathbf{x}\|_1 + \|(\mathbf{z}-\mathbf{x})_S\|_p.$$

In view of inequality (2.6), we get

$$\begin{aligned}\|\mathbf{z}-\mathbf{x}\|_p &\leq \frac{1}{s^{1-1/p}}\|\mathbf{z}-\mathbf{x}\|_1 + \frac{\rho}{s^{1-1/p}}\|(\mathbf{z}-\mathbf{x})_{\overline{S}}\|_1 + \tau s^{1/p-1/q}\|A(\mathbf{z}-\mathbf{x})\| \\ &\leq \frac{1+\rho}{s^{1-1/p}}\|\mathbf{z}-\mathbf{x}\|_1 + \tau s^{1/p-1/q}\|A(\mathbf{z}-\mathbf{x})\|. \end{aligned}\tag{2.8}$$

Substituting (2.7) into the above inequality (2.8), gives us

$$\begin{aligned}\|\mathbf{z}-\mathbf{x}\|_p \leq \frac{(1+\rho)^2}{(1-\rho)}\frac{1}{s^{1-1/p}}(\|\mathbf{z}\|_1 - \|\mathbf{x}\|_1 + 2\sigma_s(\mathbf{x})_1) \\ +\frac{(3+\rho)}{(1-\rho)}\tau s^{1/p-1/q}\|A(\mathbf{z}-\mathbf{x})\|.\end{aligned}$$

Setting $C := (1+\rho)^2/(1-\rho)$ and $D := (3+\rho)\tau/(1-\rho)$ and substituting for it above gives us the desired inequality from Theorem 2.17.

■

2.5 Matrix Analysis: Coherence

It's not easy to verify if a certain matrix is suitable for Basis Pursuit, that is, to check if the given matrix has the NSP. Thus it would be practical if we could find an easy way to calculate a quantity from which we can guarantee the success of the recovery algorithm. The *coherence* is such a quantity.

Definition 2.19. Let $A \in \mathbb{C}^{m\times N}$ be a matrix with l_2-normalized columns, $\mathbf{a}_1, ..., \mathbf{a}_N$ such that $\|\mathbf{a}_i\|_2 = 1$ for all $i \in [N]$. The *coherence* $\mu = \mu(A)$ of the matrix A is defined as

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

The l_1-*coherence* μ_1 is defined for $s \in [N-1]$ as

$$\mu_1(s) := \max_{i\in N} \max\left\{\sum_{j\in S} |\langle \mathbf{a}_i, \mathbf{a}_j\rangle|, S \subset [N], \text{card}(S) = s, i \notin S\right\}$$

From the definition of coherence we have the following result:

Theorem 2.20. *Let $A \in \mathbb{C}^{m\times N}$ be a matrix with l_2-normalized columns. If*

$$\mu_1(s) + \mu_1(s-1) < 1,$$

then every s-sparse vector $\mathbf{x} \in \mathbb{C}^N$ is exactly recovered from the vector $\mathbf{y} = A\mathbf{x}$ via Basis Pursuit.

Proof. If we can show that for a given matrix $A \in \mathbb{C}^{m\times N}$, with l_2-normalized columns satisfying $\mu_1(s) + \mu_1(s-1) < 1$ also possess the NSP of order s, then according to Theorem 2.10, every s-sparse vector $\mathbf{x}$ recovered via Basis Pursuit, is unique.
Let $\mathbf{a}_1, ..., \mathbf{a}_N$ denote the columns of A. If $\mathbf{v} \in \ker(A)$, then $A\mathbf{v} = \mathbf{0}$ can be rewritten as the sum of the column vectors of A as the sum $\sum_{j=1}^{N} v_j \mathbf{a}_j = \mathbf{0}$. Since the $\mu_1(s)$ coherence function is a statement about the sum of the inner products of pairwise columns of A, it will be convenient to rewrite a particular weight v_i as:

$$v_i = v_i\langle \mathbf{a}_i, \mathbf{a}_i\rangle = -\sum_{j=1, j\neq i}^{N} v_j \langle \mathbf{a}_j, \mathbf{a}_i\rangle$$

Splitting $[N]$ into two parts S and $\overline{S}$ and taking the absolute value of v_i it follows that

$$|v_i| \leq \sum_{l\in\overline{S}} |v_l||\langle \mathbf{a}_l, \mathbf{a}_i\rangle| + \sum_{j\in S, j\neq i} |v_j||\langle \mathbf{a}_j, \mathbf{a}_i\rangle|$$

Summing over all $i \in S$ we get

$$\|\mathbf{v}_S\|_1 = \sum_{i\in S}|v_i| \leq \sum_{l\in\overline{S}}|v_l|\sum_{i\in S}|\langle\mathbf{a}_l,\mathbf{a}_i\rangle| + \sum_{j\in S}|v_j|\sum_{i\in S, i\neq j}|\langle\mathbf{a}_j,\mathbf{a}_i\rangle|$$

$$\leq \sum_{l\in\overline{S}}|v_l|\mu_1(s) + \sum_{j\in S}|v_j|\mu_1(s-1) = \mu_1(s)\|\mathbf{v}_{\overline{S}}\|_1 + \mu_1(s-1)\|\mathbf{v}_S\|_1$$

Rearranging and neglecting the terms between the first term and the last in the equations above we get

$$(1-\mu_1(s-1))\|\mathbf{v}_S\|_1 \leq \mu_1(s)\|\mathbf{v}_{\overline{S}}\|_1$$

using the first part of the condition in the theorem, $\mu_1(s) + \mu_1(s-1) < 1$, we get the desired inequality

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

which concludes the proof.

■

CHAPTER 3

Neural Networks And Deep Learning

Neural Networks (NN's), a biologically-inspired programming paradigm, provide among the best solutions to many problems where the neural network has a lot of training data. Consequently, NN's perform well in areas such as image classification, speech recognition, natural language processing, and so forth. However, NN's have reportedly seen to suffer from instabilities in image reconstruction. Our goal is to get mathematical insight to why these instabilities occur. We begin by introducing the basic elements of NN's in framework of supervised machine learning.

3.1 Supervised Machine Learning

In supervised machine learning, the objective is to learn a mapping f from an input space $\mathbb{U}$ to an output space $\mathbb{V}$. The 2-tuple set $\{(u_i, v_i)\}_{i=1}^n$, $u_i \in \mathbb{U}$, $v_i \in \mathbb{V}$ is refer to as the *training set*. By varying the output space $\mathbb{V}$, we can choose what our function f should model.
For instance, if v_i takes the values -1 or 1 , then f is modelling a *binary classification* problem or if v_i takes three or more discrete values then f is modelling the *multiclass classification* problem.

Image reconstruction can be cast as a supervised learning problem as following : Let $\mathbb{U} \subseteq \mathbb{R}^M$ and $\mathbb{V} \subseteq \mathbb{R}^N$ such that $\{(y_i, x_i)\}_{i=1}^n$ is the training set with x_i as images and y_i the corresponding measurements. Our goal is then to find a matrix A such that we can express

$$y_i = Ax_i + \epsilon, \quad i \in \{1, ..., n\}$$

which corresponds to learning the mapping $f(u_i) = v_i + \epsilon$, $i \in \{1, ..., n\}$. It is desirable that this mapping has *small training error*, i.e. $f(u_i) \approx v_i$ for $i =$ 1,...,n, but also that the mapping *generalizes* well to new data $\{(y_k, x_k)\}$ which is close to , but not part of the training set.
The task of computing the mapping f is known as *training the model*, such a mapping could for instance be a NN.

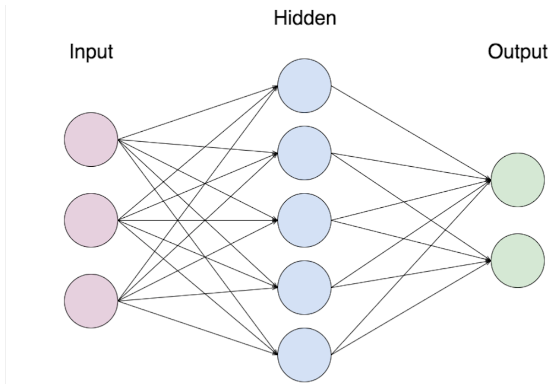

Figure 3.1: Extremely simplified NN with a single hidden layer.

3.2 Design

The term neural network covers a large class of models and learning methods. Here we try to give a description that is sufficient for the later analysis.

Definition 3.1. Let $n_0, ..., n_L \in \mathbb{N}$. Let $W_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ for $l = 1, 2, ..., L$ be affine maps. Let $\rho_1, ..., \rho_{L-1} : \mathbb{R} \to \mathbb{R}$ be non-linear, differentiable functions and let $n_0 = M$ and $n_L = N$. Then a map $\Psi : \mathbb{R}^M \to \mathbb{R}^N$ given by

$$\Psi(y) = W_L(\rho_{L-1}(...\rho_1(W_1(y))...))$$

is called a *Neural Network*.

The type of network defined above is a feedforward network. It takes an input y and *feeds* it forward through the network via an alternating sequence of affine maps and non-linear activation functions. It is common to visualize the network as a graph, where the nodes of the graph are the *neurons* and the entries of W_l are the weights assigned to each edge of the graph. At a given *layer*, each neuron takes in a sum of linear combination from the outputs of the neurons in the previous layer, applies a bias, applies an activation function and finally feeds it forward to the next layer of neurons. The number of layers and the number of neurons per layer together with the the activation function ρ is called the *architecture* of the network.

The choice of activation function may be important to the performance of the network. Common activation functions in the literature include:

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$
$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

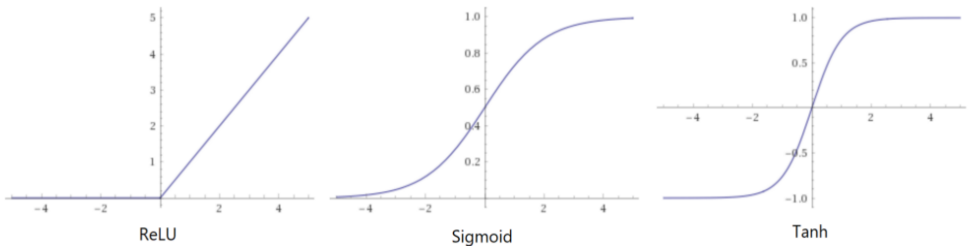

Figure 3.2: Different choices of activations functions.

If we look closer at the sigmoid function and the hyperbolic tangent, we observe that for large negative input or large positive input , the derivative of these functions will be close to zero. This is known as the *vanishing gradient problem*, which can cause the *training* of the network to be slow, which is the topic of the next section.

3.3 Training

The training of a NN is to find values for the weights and biases such that the NN performs well on a given data set. To simplify notation, we will let θ be defined as the set of all trainable parameters. For a neural network this will typically be:

$$\theta = \left\{\mathbf{W}_1, ..., \mathbf{W}_L, \mathbf{b}_1, ..., \mathbf{b}_L\right\}$$

We write Ψ_θ to emphasize that this NN uses θ as its parameters.

One way to measure how well the network is trained, is by evaluating its performance on a given set by a *cost function*.

Cost function

Several cost functions may be defined on the network. A popular choice is the *quadratic* cost function, also known as the *mean squared error(MSE)* given by:

$$C(\theta|(\mathbf{u}_1, \mathbf{v}_1), ..., (\mathbf{u}_n, \mathbf{v}_n)) = \frac{1}{2n}\sum_{i=1}^{n} \|\Psi(\mathbf{u}_i) - \mathbf{v}_i\|^2 \tag{3.1}$$

Here n is the total number of samples, $\mathbf{u}_i$ the input sample into the network and $\mathbf{v}_i$ the corresponding validation sample. Next, we want to optimize the trainable parameters θ, that is to solve:

$$\text{minimize} \quad C(\theta|(\mathbf{u}_1, \mathbf{v}_1), ..., (\mathbf{u}_n, \mathbf{v}_n)) \tag{3.2}$$

This problem is nonlinear, nonconvex and there is generally no possibility of computing global minimizers of it. Even worse, since a NN needs a huge number of parameters to perform well and a large set of training data, it is a major computational challenge.

Gradient Descent Methods

To make the training of the NN's practical regarding the computational demand, a first-order optimization method is usually used to solve (3.2). These are methods based on the gradient descent. In a gradient descent method we pick a starting point w_0, then we follow the path of steepest descent, that is, we produce the sequence:

$$w_{i+1} = w_i - \tau_i \nabla C(w_i), \ i = 0, 1, ... \tag{3.3}$$

Calculating the gradient requires one pass through all the training data. When the amount of training data is large, this ceases to be computationally feasible. However, there is at least one way to reduce the number of computational steps, known as *Stochastic Gradient Descent*.

There are several variations, but the main idea is to compute the gradient from a smaller subset of the training data and thereby reducing the computation needed at step i.

In order to apply the stochastic gradient descent method, we need to compute the gradient. The common algorithm for computing such gradients, is known as *backpropagation*.

3.4 The Universal Approximation Theorem

In supervised machine learning we are assuming that there exists an underlying function that we are approximating. The Universal Approximation Theorem gives justification for the use of NN's to approximate such a function.

Theorem 3.2. *(Universal Approximation Theorem) Let $\mathcal{S} \subset \mathbb{R}^n$ be a compact set and let $\mathcal{C}(\mathcal{S})$ denote the vector space of continuous functions on $\mathcal{S}$. Let $\sigma \in \mathcal{C}(\mathcal{S})$.*
Then for any $g \in \mathcal{C}(\mathcal{S})$ and any $\epsilon > 0$, there exists a set of parameters $n \in \mathbb{N}$, $c_1, ..., c_n \in \mathbb{R}$, $\mathbf{w}_1, ..., \mathbf{w}_n \in \mathbb{R}^n$, $b_1, ..., b_n \in \mathbb{R}$ such that $f : \mathbb{R}^n \to \mathbb{R}$ defined as

$$f(\mathbf{x}) = \sum_{i=1}^{n} c_i \sigma(\mathbf{w}_i^T \mathbf{x}_i + b_i) \tag{3.4}$$

satisfies

$$\|f(\mathbf{x}) - g(\mathbf{x})\| < \epsilon$$

for all $\mathbf{x} \in \mathcal{S}$ if and only if σ is not a polynomial.

In other words, the class of neural networks with one hidden layer are dense in $\mathcal{C}(\mathcal{S})$.

Proof. If σ is a polynomial of degree m, then for every choice of $\mathbf{x} \in \mathbb{R}^n$ and $b \in \mathbb{R}$, $\sigma(\mathbf{w}^T\mathbf{x} + b)$ is polynomial of total degree at most m, and thus does not span $\mathcal{C}(\mathcal{S})$. For the converse, see ■

her trengs referanse

The Universal Approximation Theorem states that the class of NNs with one hidden layer is already extremely rich. However, there are reasons for considering deeper NN's, that is with more layers, since they might approximate certain functions more efficiently than shallow NNs.

3.5 The Success Of Deep Learning

trengs ref til ibm

According to IBM, Deep Learning is a subset of machine learning, which is essentially a NN with three or more layers. As we have seen in this chapter, solving a supervised machine learning problem with Deep Learning involves choosing a suitable NN architecture, optimization algorithm and hyperparameters, and then train the NNs large numbers of parameters by feeding it data.
Deep Learning has achieved a great number of successes during the last decade, yielding state-of-the-art performance on a range of challenging machine learning problems. Since the list of successes in different problems is quite long, we will only point out two examples that are milestones in history of Deep Learning applied to image classification tasks. The first milestone was achieved in 2012, when a Deep Learning-based classifier named *AlexNet* won the *ImageNet Large Scale Visual Recognition Challenge(ILSVRC)*. Prior to 2012, the winners were not Deep Learning based. Not only did it win, but it also achieved an error reduction of nearly 10% over the previous year's winner. Since then, all winners of the ILSVRC contest have used deep learning. In less than a decade, the error rate was reduced from 30% to less than 4%. In 2015, Deep Learning reached a second milestone, when it reduced the image classification error rate to sub 5%, which is the error rate incurred by humans. As a result of this performance, the term 'superhuman' has been used to describe Deep Learning's performance on image classification. To the authors knowledge, deep learning is now the method of choice for most image classification tasks.

3.6 Instabilities In Deep Learning

Even though Deep Learning has great success in image classification problems, its performance might also be unstable, particularly against small perturbations in their input. Given a trained classifier C and an input image x_0, it is possible to find a visually imperceptible perturbation r for which the network misclassify the image $x_0 + r$ such that

$$C(x_0) \neq C(x_0 + r)$$

There even exist methods for finding perturbation that purposefully achieves the misclassification effect. One such method is known as *DeepFool*. We omit the details of the method, but note that around 2014, DeepFool was used to find a small perturbation, that was applied to the previously mentioned $ImageNet$ dataset of images, such that the perturbation achieved a classification error rate of over 75 %. Making matters even worse, the perturbation are universal and transferrable, meaning that the same perturbation can be applied to a set of images, and consequently fool, i.e. misclassify, several NNs.

Why do instabilities occur?

With the existence of visually imperceptible changes to an image that fools a NN one may wonder what Deep Learning actually learns. It would be reasonable to assume that humans perceive the abstract characteristics of everyday life images, in a way that is more resilient to imperceptible changes, compared to the current versions of NNs. Some insight might be gain from the so-called *false structure* phenomenon. We will give an example illustrating the phenomenon in shallow NNs:

Consider the set of 3×3 images with a light stripe that is either horizontal or vertical. Let U denote the collection of all such images where, in the vertical case, the light stripe takes the value $1 + a$ and the dark stripes take the value $1 - a$ for some $0 \leq a \leq 1/4$. In the horizontal case, the light stripe takes the value $1 - a$ and the dark stripes take the value $-a$. Next, define the binary labelling function

$$C_0(x) = \begin{cases} +1 \text{ if x has a horizontal light stripe,} \\ -1 \text{ if x has a vertical light stripe.} \end{cases}$$

Now consider a one-layer NN which tries to approximate the above function C_0 defined by

$$C(x) = \begin{cases} +1 \text{ if the sum of the pixel values are} \leq 3, \\ -1 \text{ if the sum of the pixel values are} > 3. \end{cases}$$

By the way it is designed, this NN a high success rate and only fails on the case $a = 0$. However, a small perturbation to the pixel values of the input image can cause C to give the wrong answer, even though the underlying ground truth function C_0 is stable regarding such a perturbation. Making the perturbation small enough, it can be invisible to the human eye.
This simple example illustrates that even though the NN may be successful, it does not capture the true underlying structure that defines the function, namely, the stripe. Thus it has learned a false structure and has become unstable to small perturbations. Knowledge of this phenomenon, might help constructing better NNs in the future.

3.7 Reasons To Go Beyond Compressed Sensing

With the success of Deep Learning in image classification problems and others, one may want to know how well NN's would perform on inverse problems such as image reconstruction compared to more well established techniques such as Compressed Sensing. Other than trying to find the upper limit on the accuracy of NN's in image reconstruction, from the fewest possible measurements, there are other reasons for exploring the methods of Deep Learning. There are at least three reasons:

i) Given a model, such as sparsity and compressibility, Compressed Sensing requires one to handcraft a reconstruction procedure, such as

l_1-minimization, that exploits this structure. It may be challenging to craft an effective reconstruction procedure for an arbitrary image model, therefore a method that is simpler to apply, might be advantageous.

ii) Optimization-based recovery procedures might be slow.

iii) Finding the optimal parameters of an iterative optimization solver is delicate, and can have a significant effect on reconstruction quality.

Once the NN is trained for image reconstruction, it could be faster than iterative optimization-based procedures, since reconstruction only requires a single forward propagation through the network. Also, the time expensive computations may be done at a more convenient time, than optimization-methods.

3.8 Deep Learning for inverse problems

With the aim of understanding the limits of Deep Learning in inverse problems, we end this chapter with a general description of how NNs may solve the discrete, linear inverse problem.

Consider the problem of recovering an unknown vector $\mathbf{x} \in \mathbb{C}^N$ from measurements $\mathbf{y} = A\mathbf{x} + \epsilon \in \mathbb{C}^m$, where $A \in \mathbb{C}^{m \times N}$. Even though most of the examples presented in this thesis have been from signal and image reconstruction, the theoretical results investigated should also apply to the abstract setting.
To apply Deep Learning to this problem, we assume there is a training set

$$\mathcal{T} = (y_i, x_i)_{i=1}^K \subset \mathbb{C}^m \times \mathbb{C}^N, y_i = Ax_i,$$

consisting of images x_i and their measurements y_i. The goal is to feed this data to the NN for it to implicitly learn a reconstruction map $\Psi : \mathbb{C}^m \to \mathbb{C}^N$ that performs well on the images of interest.

CHAPTER 4

Instabilities In Deep Learning For Inverse Problems

In many fields the perils and limitations of Deep Learning used in inverse problems have become an issue. In particular, the issue of instability and NN-generated "hallucinations" across many image modalities is a rising concern. An example is from the Facebook fastMRI challenge(2020) :

"Such hallucinatory features are not acceptable and especially problematic if they mimic normal structures that are either not present or actually abnormal. Neural network models can be unstable as demonstrated via adversarial perturbation studies."

Another example is from super resolution techniques, which can be viewed as an undersampled inverse problem:

"However, if the neural network encounters unknown specimens, or known specimens imaged with unknown microscopes, it can produce nonsensical results ."

which is from "The promise and peril of deep learning in microscopy", *Nature Methods(2021).*

A final example is from process of outlining a legal framework for the use of AI (Deep Learning):

"On AI, trust is a must, not a nice to have. [...] The new AI regulation will make sure that Europeans can trust what AI has to offer. [...] High-risk AI systems will be subject to strict obligations before they can be put on the market: [requiring] High level of robustness, security and accuracy."

which is from *The European Commissions* "Outline for legal AI" (April 2021).

With the additional motivation of these examples we begin our study of accuracy and stability of NNs with the introduction of a key theoretical result.

4.1 Lack Of Kernel Awareness

Suppose we have trained a NN to approximate an inverse map, $\Psi : \mathbb{C}^m \to \mathbb{C}^N$ where $m << N$. For the NN to be useful, it should perform well on the training set and test set. However, if the NN recovers two vectors $\mathbf{x}, \mathbf{x}'$ whose difference lies in the null space of A, then there exist a ball around $\mathbf{y} = A\mathbf{x}$ where the NN may perform badly with small perturbations from $\mathbf{y}$. Before we state this consequence as a theorem, recall that local ϵ-Lipschitz constant of a function ϕ at $\mathbf{y} \in \mathbb{C}^m$ is defined as

$$L^\epsilon(\phi, y) = \sup_{0<d_2(z,y,)\leq\epsilon} \frac{d_1(\phi(z), \phi(y))}{d_2(z, y)}$$

Theorem 4.1. *(Universal Instability Theorem) Let d_1 and d_2 be metrics on $\mathbb{C}^N$ and $\mathbb{C}^m$ respectively, $A : \mathbb{C}^N \to \mathbb{C}^m$ a linear map, and $\Psi : \mathbb{C}^m \to \mathbb{C}^N$ a continuous reconstruction map. Suppose there exist x, x' $\in \mathbb{C}^N$ and $\eta > 0$ such that*

$$d_1(\Psi(Ax), x) < \eta, \;\; d_1(\Psi(Ax'), x') < \eta, \tag{4.1}$$

$$d_2(Ax, Ax') \leq \eta. \tag{4.2}$$

Then there is a closed non-empty ball $\mathcal{B} \subset \mathbb{C}^m$ centred at $y = Ax$ such that the local ϵ-Lipschitz constant at any $\tilde{y} \in \mathcal{B}$ is bounded from below:

$$L^\epsilon(\Psi, \tilde{y}) \geq \frac{1}{\epsilon}(d_1(x, x') - 2\eta), \quad \epsilon \geq \eta. \tag{4.3}$$

Proof. By the definition of the supremum we have that

$$L^\epsilon(\Psi, y) = \sup_{0<d_2(z,y,)\leq\epsilon} \frac{d_1(\Psi(z), \Psi(y))}{d_2(z, y)} \geq \frac{d_1(\Psi(Ax), \Psi(Ax'))}{d_2(Ax, Ax')}$$

Applying the reverse triangle inequality twice we get

$$\frac{d_1(\Psi(Ax), \Psi(Ax'))}{d_2(Ax, Ax')} \geq \frac{d_1(x, \Psi(Ax')) - d_1(\Psi(Ax), x)}{d_2(Ax, Ax')} \geq \frac{d_1(x, x') - d_1(\Psi(Ax), x) - d_1(\Psi(Ax'), x')}{d_2(Ax, Ax')}$$

Applying assumption (4.1) and making the bound sharper with $\epsilon \geq \eta$, we get

$$\frac{d_1(x, x') - d_1(\Psi(Ax), x) - d_1(\Psi(Ax'), x')}{d_2(Ax, Ax')} > \frac{d_1(x, x') - 2\eta}{\epsilon}$$

Neglecting the middleterms we get

$$L^\epsilon(\Psi, y) > \frac{d_1(x, x') - 2\eta}{\epsilon}.$$

Next, let $\eta_1 = d_1(\Psi(Ax), x)$, and observe that $\eta_1 < \eta$ by (4.1).
Since Ψ is continuous at $y = Ax$, there exist a $\delta > 0$ such that for all $\tilde{y} \in \mathbb{C}^m$ with $d_2(y, \tilde{y}) < \delta$, we have that $d_1(\Psi(y), \Psi(\tilde{y})) \leq \eta - \eta_1$. Thus, for

a specific $\delta > 0$, we get that $d_1(\Psi(y), \Psi(\tilde{y})) \leq \eta - \eta_1$ for all $\tilde{y} \in \mathcal{B}_1$, where $\mathcal{B}_1 = \{\tilde{y} \in \mathbb{C}^m : d_2(y, \tilde{y}) < \delta)\}$. Next, let $\mathcal{B}_2 = \{\tilde{y} \in \mathbb{C}^m : d_2(Ax', \tilde{y}) \leq \eta)\}$ and $\mathcal{B} = \mathcal{B}_1 \cap \mathcal{B}_2$. Observe that $\mathcal{B} \neq \emptyset$, since $y \in \mathcal{B}$. Thus, for any $\tilde{y} \in \mathcal{B}$ we have

$$\begin{aligned}
L^\epsilon(\Psi, \tilde{y}) &= \sup_{0<d_2(z,\tilde{y},)\leq\epsilon} \frac{d_1(\Psi(z), \Psi(\tilde{y}))}{d_2(z, \tilde{y})} \geq \frac{d_1(\Psi(Ax'), \Psi(\tilde{y}))}{d_2(Ax', \tilde{y})} \\
&\geq \frac{d_1(\Psi(Ax), \Psi(Ax')) - d_1(\Psi(Ax), \Psi(\tilde{y}))}{d_2(Ax', \tilde{y})} \\
&\geq \frac{d_1(x, x') - d_1(\Psi(Ax), x) - d_1(\Psi(Ax'), x') - d_1(\Psi(Ax), \Psi(\tilde{y}))}{d_2(Ax', Ax)} \\
&\geq \frac{d_1(x, x') - \eta_1 - \eta - (\eta - \eta_1)}{d_2(Ax', Ax)} \\
&\geq \frac{1}{\epsilon}(d_1(x, x') - 2\eta).
\end{aligned}$$

which is the desired inequality (4.3).

■

Appendices