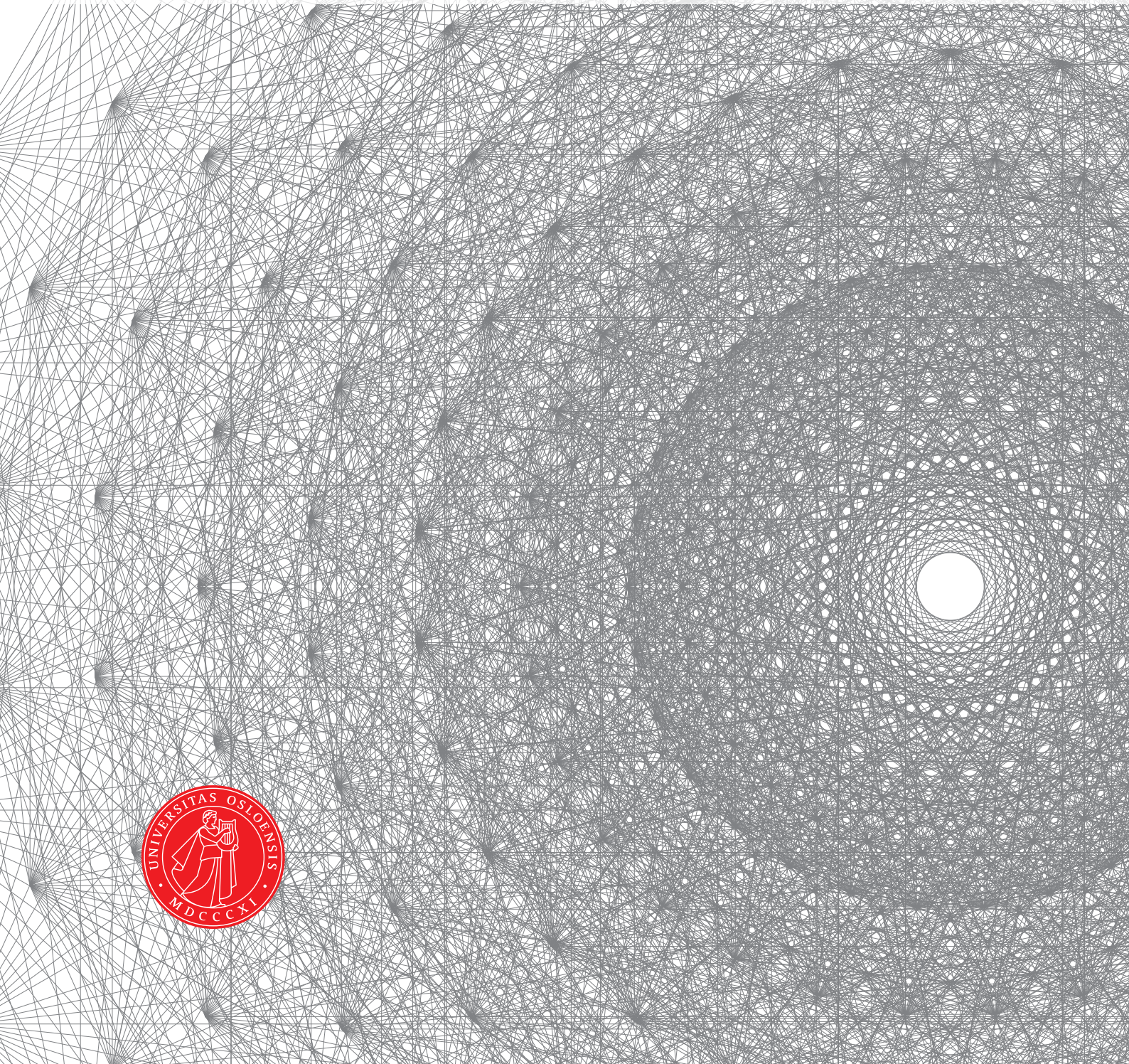
UiO : Department of Mathematics
University of Oslo

Title

Optional Subtitle

Andras Filip Plaian

Master's Thesis, Autumn 2021



The study programme is unspecified. Please consult the documentation for the package `masterfrontpage` in order to correctly print the colophon.

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

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

Informally, a vector is compressible if $l_p - error$ 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:*

(i) *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 $\mathbf{z} = \mathbf{x}$.*

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

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

(iv) *Every set of 2s columns of A is linearly independent.*

Proof. $i) \implies ii)$ ■

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:

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

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 general $A \in \mathbb{C}^{m\times N}$ and $\mathbf{y} \in \mathbb{C}^m$ 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 class.

Proof. 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 $[m]$? Let $\{\mathcal{C}_i, i \in [N]\}$ be the collection of 3-element subsets of $[m]$. Define vectors

■

P_0 optimization belongs to a class of NP-hard problems, meaning that any direct approach in solving P_0 is intractable. Before proving

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.

Basis Pursuit

In order to show why BP can solve (P_0) we need to introduce the 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 $|S| \leq s$

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

From the results above it is clear that if a matrix possesses the NSP property of order s, the BP will solve $(P1)$. 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 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 $card(S)$=$2s$, the matrix A_S 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 *has to satisfy:*

$$2s \leq rank\ A \leq m$$

Stability And Robustness

Basis Pursuit features two important properties, namely *stability* and *robustness*. These two properties tackle sparsity defects and noise .

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

$$\|\mathbf{v}_S\|_1 < \rho\|\mathbf{v}_{\overline{S}}\|_1 + \tau\|A\mathbf{v}\| \text{ for all } \mathbf{v} \in \mathbb{C}^N$$

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

The following result....

Theorem 2.14. *Given* $1 \leq p \leq q$*, suppose that the matrix* $A \in \mathbb{C}^{M\times N}$ *satisfies the robust null space property of order* s*. Then, for any* $\mathbf{x}, \mathbf{z} \in \mathbb{C}^N$*,*

$$\|\mathbf{x} - \mathbf{z}\|_p \leq$$

Proof. Her trengs bevis. ■

CHAPTER 3

Neural Networks

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 $\mathbb{U} \subseteq \mathbb{C}^M$ and $\mathbb{V} \subseteq \mathbb{C}^N$ 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.

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.

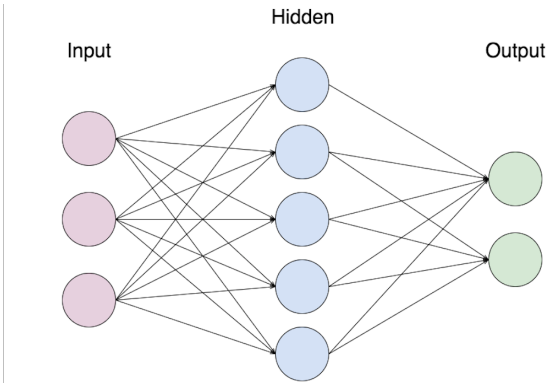


Figure 3.1: Extremely simplified NN with a single hidden layer.

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

$$\phi(y) = \rho_L(W_L(...\rho_1(W_1(y))...))$$

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:

$$ReLU(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

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*, we shall refer to it in the next section.

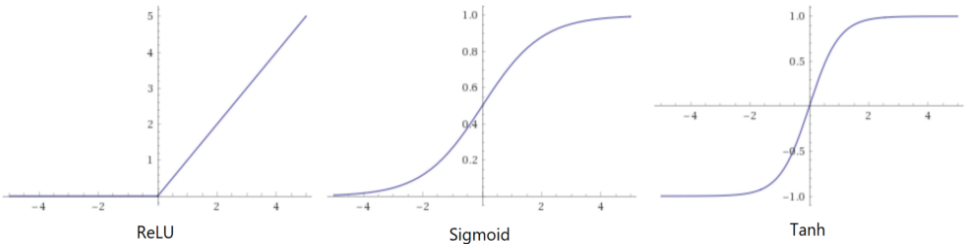


Figure 3.2: Different choices of activations functions.

3.3 Training

The first step in training a NN is to find weights and biases such that the network performs well on the training set. A way to measure how well the network is performing is by defining a *cost function*.

Loss function

Several cost functions may be defined, a popular choice is the *quadratic* cost function, also known as *mean squared error(MSE)*:

$$C(w) = \frac{1}{2n}\sum_{i=1}^{n}\|\phi(u_i) - v_i\|^2 \tag{3.1}$$

Here, w denotes the collection of all weights and biases in the network, n is the total number of training inputs, v_i is the vector of outputs from the network when u_i is input, and the sum is over all training inputs, u_i. Next, we want to optimize w and b, that is to solve:

$$\min C(w) \tag{3.2}$$

This problem is nonlinear, nonconvex and there is generally no possibility of computing global minimizers of it. Even worse, since a NN needs alot of parameters to perform well and alot of training data, it is a major computational challenge. However, there is a practical way to reduce the computational cost, known as *Stochastic Gradient Descent*.

Stochastic Gradient Descent

To make NN's training practical regarding the computational demand, a first-order optimization method is used to solve (ref). These are based on 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 set. When the amount of training data is large, this ceases to be computationally feasible. One alternative is already mentioned, 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 $C(w_i)$ including all the weights in the network. 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. Before we state the theorem, we recall the definition of a *discriminatory* function.

Definition 3.2. Let $M(S)$ be the space of finite, signed Borel measures on the compact set S. We say ρ is *discriminatory* if for $\mu \in M(S)$ and

$$\int_{I_n} \rho(\mathbf{w}^\mathbf{T}\mathbf{x} + b)\ d\mu(x) = 0$$

for all $\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}$ then $\mu = 0$.

We are now ready to present the Universal Approximation Theorem and its proof.

Theorem 3.3. *(Universal Approximation Theorem) Let $C(S)$ be the space of continuous functions on S.*
If the activation function ρ in the neural network definition is a continuous, discriminatory function, then the set of all neural networks $\mathcal{N}$ is dense in $C(S)$.

Proof. To prove that $\mathcal{N}$ is dense in $C(S)$, we will prove that its closure is $C(S)$. By way of contradiction, suppose that $\overline{\mathcal{N}} \neq C(S)$, then $\overline{\mathcal{N}}$ is a closed, proper subspace of $C(S)$. By the Hahn-Banach Theorem, there exist a bounded linear functional L, such that $L(\mathcal{N}) = L(\overline{\mathcal{N}}) = 0$ but $L \neq 0$.

By the Riesz Representation Theorem, we can write the functional L as

$$L(h) = \int_S h(x)\ d\mu(x)$$

for some $\mu \in M(S)$ and for all $h \in C(S)$. In particular, since by definition any NN is a member of $\mathcal{N}$, and L is identically zero on $\mathcal{N}$, we have

$$\int_S h(x)\ d\mu(x) = 0$$

Since we ρ is discriminatory, we have $\mu = 0$. This contradicts that $L \neq 0$.

Therefore, $\mathcal{N}$ is dense in $C(S)$. ■

CHAPTER 4

Instabilities In Deep Learning

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 definition of the supremum, the reverse triangle inequality (r.t.i.), assumption (4.1) and $\epsilon \geq \eta$ we get the following chain of inequalities

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

$$\overset{r.t.i.}{\geq} \frac{d_1(x, \Psi(Ax')) - d_1(\Psi(Ax), x)}{d_2(Ax, Ax')} \overset{r.t.i.}{\geq} \frac{d_1(x, x') - d_1(\Psi(Ax), x) - d_1(\Psi(Ax'), x')}{d_2(Ax, Ax')}$$

$$\overset{(4.1)}{>} \frac{d_1(x, x') - 2\eta}{d_2(Ax, Ax')} \overset{\epsilon\geq\eta}{\geq} \frac{d_1(x, x') - 2\eta}{\epsilon}$$

Finally, since Ψ is continuous, we get

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

for all $\tilde{y} \in \mathcal{B}$ where $\mathcal{B}$ is some ball around $y = Ax$.

■

noe mer her

Appendices