UiO : **Department of Mathematics**

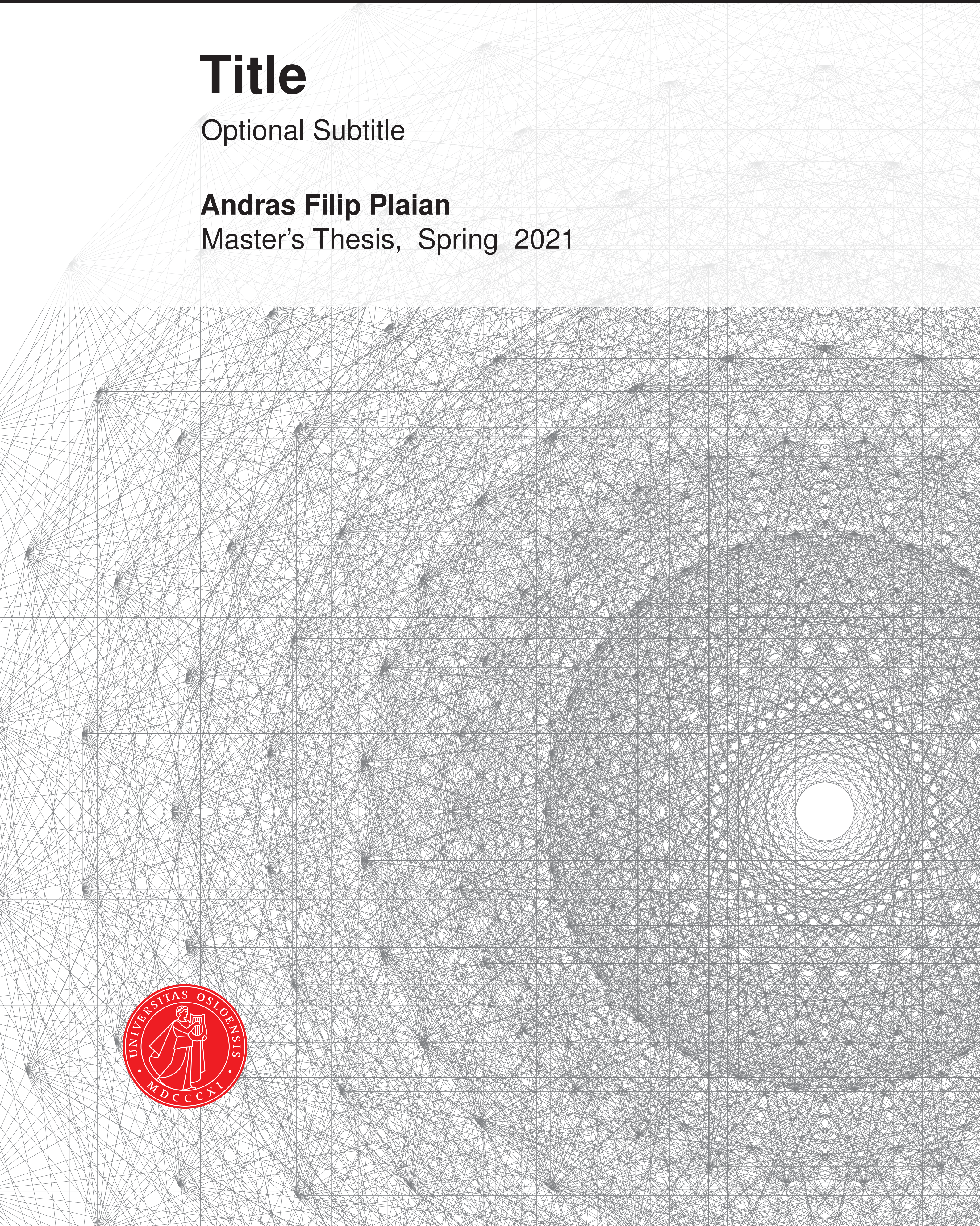
University of Oslo

# Title

Optional Subtitle

**Andras Filip Plaian**

Master's Thesis, Spring 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

# Compressive Sensing

In an underdetermined system of linear equations there are fewer equations than unknowns. In mathematical terms this can we 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 $M < N$. This equation is unsolvable in the general case, however, under certain conditions, it is possible to find exact or estimated solutions. The underlying assumptions that make it possible is sparsity and compressibility. The research area associated with these assumptions is called compressive sensing. The goal of this chapter is to show that by using compressive sensing techniques it is possible to construct a stable and robust mapping $\mathbf{B} : \mathbb{C}^M \to \mathbb{C}^N$ such that an estimate or solution would exist for the *inverse problem* $\mathbf{x} = A^{-1}\mathbf{y}$.

## 2.1 Sparsity and Compressibility

This section introduces the reader to the main assumptions and their notions in CS. Necessary terminology will be introduced appropriately.

**Definition 2.1.1.** 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 due to not satisfying the scaling property of norms. We can now define *s-sparse* vectors.

**Definition 2.1.2.** 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 our vector is only close to being sparse. In order for CS to tackle more problems, we may also be interested in the following notion of *compressibility.*

**Definition 2.1.3.** 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 $s$.

## 2.2 Algorithms

In line with our goal to construct the mapping $\mathbf{B} : \mathbb{C}^M \to \mathbb{C}^N$ it will be useful to restate the compressive sensing problem as an optimization problem and to show that by solving the compressive sensing problem, that is to find the s-sparse vector consistent with Ax=y, we also solve the map construction problem.

$$\min \|\mathbf{z}\|_0 \text{ subject to } A\mathbf{z} = A\mathbf{x} \tag{2.1}$$

$(P_0)$ is a combinatorial optimization problem, but the problem is generally NP-hard. Since this makes $(P_0)$ intractable, we will solve the convex relaxation of $(P_0)$ instead.

$$\min\|\mathbf{z}\|_1 \text{ subject to } A\mathbf{z} = A\mathbf{x} \tag{2.2}$$

Solving the inverse problem by solving $(P_1)$ is known as Basis Pursuit.

### 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 2.2.1.** 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 2.2.2.** *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 2.2.3.** *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 2.2.4.** *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 2.2.5.** 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 2.2.6.** *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.

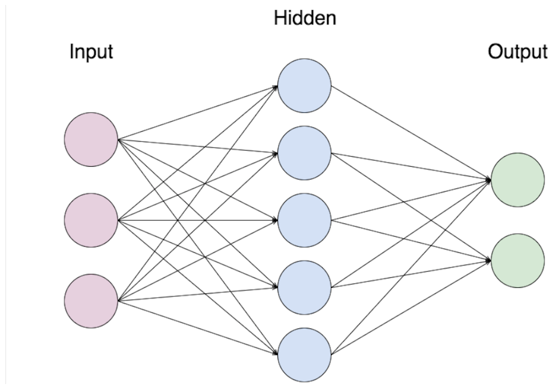


Figure 3.1: Extremely simplified NN with a single hidden layer.

**Definition 3.2.1.** 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 $\rho$ 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.

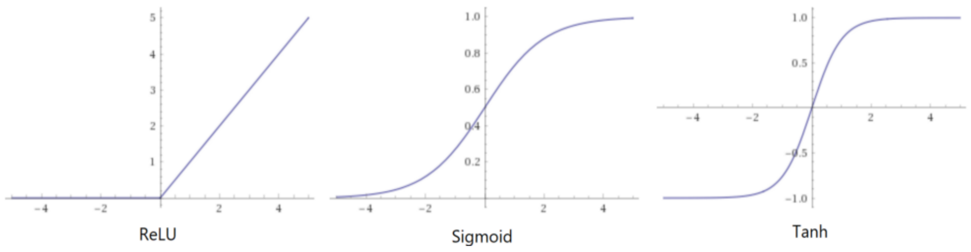


Figure 3.2: Different choices of activations functions.

## 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:

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

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 3.4.1.** Let $M(S)$ be the space of finite, signed Borel measures on the compact set $S$. We say $\rho$ 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 3.4.2.** *(Universal Approximation Theorem) Let $C(S)$ be the space of continuous functions on $S$.*
*If the activation function $\rho$ 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 $\rho$ 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 $\epsilon$-Lipschitz constant of a function $\phi$ 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 4.0.1.** *(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 $\epsilon$-Lipschitz constant at any $\tilde{y} \in \mathcal{B}$ is bounden 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 $\Psi$ 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