Study of AutoEncoders for Step Functions

February 17, 2022

1 Introduction

Autoencoders are a special type of Neural Networks that have been conceived to learn data representation. An autoencoder is composed of two connected networks called Encoder and Decoder. The Encoder is fed high dimensional input data. And it learns to compress the data into useful low dimensional representation named code or latent representation. The second network, the Decoder, takes as input the output code of the Encoder. It learns to reconstruct, from this code, its corresponding high dimensional data with minimal distortion. Thanks to this structure, the encoder can learn to extract from the data necessary and sufficient characteristics in an unsupervised fashion.

The unsupervised compression ability of the autoencoder would be ideal if each dimension of the latent code is meaningful to us humans. In other words, an ideal autoencoder would be able to summaries data, without a teacher, as we human do. Unfortunately, nowadays, this perfect world is far from true because of the following problems. First, we don't always know the correct dimension to give for the latent space, since it is preset before the learning. Second, the meaningful features are prone to overlap over the dimensions and/or to be non linear with respect to a certain dimension.

In this paper, we are interested in the linear encoding of the parameters. Therefore, we will study the behaviour of autoencoders for step functions with constant amplitude and varying positions.

2 Notations

- Scalars are written in normal font, vectors in bold and matrices are denoted by a capital letter.
- $\mathbf{x} \in \mathbb{R}^n$ is an input sample to the autoencoder. It represented a step function. In total, with a vector of dimension n we can construct $n+1$ possible step functions if we only vary the position of the transition while keeping the non-zero amplitude fixed.
- The autoencoder's output is denoted by $\mathbf{y}$. It is a vector of dimension n as well. Ideally, after training, $\mathbf{y}$ should be exactly equal to $\mathbf{x}$ that is on the input.
- $\mathbf{z}$ denotes the latent code.
- The letter $i \in 1, 2, ..., n+1$ is used to indicate a particular sample. The letter $j \in 1, 2, ..., n$ is used to indicate an entry of a vector of dimension n. For example $x_j^{(i)}$ represents the entry j of the input sample i.
- Weights of a layer l are represented by $W^{[l]}$. And biases are represented by $\mathbf{b}^{[l]}$.
- We denote a preactivation vector of a layer l corresponding to a sample i by $\mathbf{u}^{[l](i)}$. And we denote the corresponding activation vector by $\mathbf{a}^{[l](i)}$.
- An activation function or non-linearity of a layer l is denoted by $g^{[l]}$. We'll have for example, $\mathbf{a}^{[l](i)} = g^{[l]}(\mathbf{u}^{[l](i)})$.

3 Data

In this paper, we will consider the simple case of only one changing parameter between the data samples. And the step function is chosen to be the elementary signal on which we will study the behaviour of autoencoders.

Definition 3.1 (Step Function)**.** *A step function* $\mathbf{x} \in \mathbb{R}^n$ *is a signal composed of two constant pieces of amplitudes* 0 *and* $C \in \mathbb{R}^+$*. The transition from* 0 *to* C *occurs at a parameter* $\theta \in \{1, ..., n\}$ *as follows,*

$$\mathbf{x}_j = \begin{cases} 0, & \text{if } j < \theta \\ C, & \text{if } j \geq \theta \end{cases} \tag{1}$$

where $\mathbf{x}_j$ *is the entry, at the position* $j \in \{1, ..., n\}$*, of* $\mathbf{x}$*.*

Throughout this paper we will use the expression "position of $\mathbf{x}$" to designate the parameter θ of the step function $\mathbf{x}$.

The dataset is made up of $n+1$ step functions $\mathbf{x}$ of dimension n, having the same amplitude C and differing in their positions θ. Note that with $\theta \in \{1, ..., n\}$ we can compose n different step functions including a constant function identically equal to C for $\theta = 1$. One might argue that a constant function is not a step function because it doesn't have a transition. But for the sake of experimenting the abilities of the autoencoder we kept the constant function identically equal to C and added another constant function identically equal to 0. Thus, the input set X is a matrix of dimension $(n+1) \times n$ where each line $i \in \{1, ..., n+1\}$ is a step function sample denoted by $\mathbf{x}^{(i)}$. Sorting the samples according to their positions, the input matrix will have the following form

$$\underset{(n+1)\times n}{X} = \begin{bmatrix} \dots & \mathbf{x}^{(1)} & \dots \\ \dots & \mathbf{x}^{(2)} & \dots \\ \dots & \mathbf{x}^{(3)} & \dots \\ & \vdots & \\ \dots & \mathbf{x}^{(n)} & \dots \\ \dots & \mathbf{x}^{(n+1)} & \dots \end{bmatrix} = \begin{bmatrix} C & C & \dots & C & C \\ 0 & C & \dots & C & C \\ 0 & 0 & \dots & C & C \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & C \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \tag{2}$$

4 Architecture

Our approach to study the behaviour of autoencoders on elementary signals is to conceive an autoencoder that can satisfy the following conditions:

(I) The architecture should be the simplest possible. That is, we will add neurons or layers only if it is necessary.

(II) The reconstruction of the signals should be perfect.

(III) The activation functions are *ReLU*s.

(IV) Since the data points differ only in their positions, the latent space dimension is fixed to 1. Thus, the code is a scalar that is expected to represent the positions θ.

(V) The code must be linear with the positions θ.

Definition 4.1 (Perfect Reconstruction)**.** *We say that an autoencoder or a decoder is achieving a perfect reconstruction if for all* $i \in \{1, ..., n+1\}$ *the output signal* $\mathbf{y}^{(i)}$ *is exactly equal to the input signal* $\mathbf{x}^{(i)}$*.*

Let Y *be the ouput matrix where each row is a reconstructed signal* $\mathbf{y}^{(i)}$*, we define the perfect reconstruction in a matrix form as*

$$\underset{(n+1)\times n}{Y} = \begin{bmatrix} \dots & \mathbf{y}^{(1)} & \dots \\ \dots & \mathbf{y}^{(2)} & \dots \\ \dots & \mathbf{y}^{(3)} & \dots \\ & \vdots & \\ \dots & \mathbf{y}^{(n)} & \dots \\ \dots & \mathbf{y}^{(n+1)} & \dots \end{bmatrix} = \underset{(n+1)\times n}{X} = \begin{bmatrix} C & C & \dots & C & C \\ 0 & C & \dots & C & C \\ 0 & 0 & \dots & C & C \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & C \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix} \tag{3}$$

4.1 One Layer Encoder

A linear encoder can be achieved by simply summing the entry values of the input signal. This can be done with one MLP layer with constant weights and without a non-linearity. !PUT FIG! . For a sample i, $z^{(i)}$ is an affine function of $\theta^{(i)}$,

$$\underset{1\times 1}{z}^{(i)} = \underset{1\times n}{\mathbf{x}}^{(i)} \underset{1\times n}{W}^{[1]^T} + \underset{1\times 1}{b}^{[1]} \tag{4}$$

$$z^{(i)} = w^{[1]} \sum_{j=1}^{n} \mathbf{x}_j^{(i)} + b^{[1]} \tag{5}$$

$$z^{(i)} = wC(n - \theta^{(i)} + 1) + b^{[1]} \tag{6}$$

where $W^{[1]} = w[1, 1, ..., 1]$ is the weight matrix, $w \in \mathbb{R}$ is a constant and $b^{[1]}$ is the bias.

Once we ensured that we have a linear encoder, we must build a decoder that can correctly reconstruct the step functions when fed the scalar codes z.

4.2 One Layer Decoder

We start by exploring the possibility of having a one layer decoder. The equations of a one MLP layer linking the code $z^{(i)}$ to the output signal $\mathbf{y}^{(i)}$ are the following,

$$\underset{1\times n}{\mathbf{u}}^{[2](i)} = \underset{1\times 1}{z}^{(i)} \underset{n\times 1}{W}^{[2]^T} + \underset{1\times n}{b}^{[2]} \tag{7}$$

$$\mathbf{y}^{(i)} = g^{[2]}(\mathbf{u}^{[2](i)}) \tag{8}$$

where $W^{[2]}$ and $\mathbf{b}^{[2]}$ are respectively the weight matrix and the bias. $\mathbf{u}^{[2]}$ is the preactivation vector of a sample i and $g^{[2]}$ is a non-linearity.

Lemma 4.1. *The one layer decoder is able to generate the $n + 1$ step functions having different positions out of their scalar codes z only if there exist two non-overlapping proper intervals A and B such that*

$$g^{[2]}(u) = \begin{cases} 0, & \text{if } u \in A \\ C, & \text{if } u \in B \end{cases} \tag{9}$$

Proof. A perfect reconstruction defined in (4.1) is equivalent to, $\forall i \in \{1, ..., n+1\}$ and $\forall j \in \{1, ..., n\}$

$$\mathbf{y}_j^{(i)} = g^{[2]}(\mathbf{u}_j^{[2](i)}) = g^{[2]}(z^{(i)} W_j^{[2]} + \mathbf{b}_j^{[2]}) = \mathbf{x}_j^{(i)} \tag{10}$$

where $x_j^{(i)} \in \{0, C\}$ and $W_j^{[2]}$ is the entry at j of $W^{[2]}$.

The code $z^{(i)}$ represents and identifies a step function according to its position. Thus $z^{(i)}$ is unique for each $\mathbf{x}^{(i)}$. Subsequently, for a fixed j and varying i, the non-linearity $g^{[2]}$ is applied on $n + 1$ different $\mathbf{u}_j^{[2](i)}$ with images in $\{0, C\}$.

More precisely, using (2) and for a fixed j, $g^{[2]}(\mathbf{u}_j^{[2](i)}) = \mathbf{x}_j^{(i)} = C$ requires the existence of a set

$$S_j^C = \{u \in \mathbb{R} : g^{[2]}(u) = C\}$$

of cardinal $|S_j^C| = j$. Similarly, $g^{[2]}(\mathbf{u}_j^{[2](i)}) = \mathbf{x}_j^{(i)} = 0$ requires the existence of a set

$$S_j^0 = \{u \in \mathbb{R} : g^{[2]}(u) = 0\}$$

of cardinal of $|S_j^0| = n + 1 - j$. Consequently, the perfect reconstruction requires the existence of two intervals $A \supset \bigcup_{j=1}^n S_j^0$ and $B \supset \bigcup_{j=1}^n S_j^C$ satisfying (9). With $g^{[2]}$ a well-defined function, A and B should be non-overlapping. Moreover, the sets S_j^C and S_j^0 are not singletons respectively for $j > 1$ and $j < n$. Thus, for $n > 1$, A and B are both proper. □

The lemma 4.1 sets a necessary condition to the perfect reconstruction using one layer decoder and a scalar code, that restricts the set of admissible activation functions. As a result a non-linearity is required at the output of the decoder. And among the well-known activation functions, only a simple threshold defined by $g^{[2]} : \mathbb{R} \to \{0, C\}, u \mapsto C\mathbb{1}_{u>0}$ satisfies this condition. However a threshold is not differentiable and thus can't be used in backpropagation. On the other hand, *ReLU* doesn't satisfy lemma 4.1 because of its linear part. As for the *sigmoid* and the *tanh*, they are strictly increasing and thus can't satisfy the condition. However, a *sigmoid* approaches 1 asymptotically for large positive values and 0 for large negative values. Therefore, we can study the use of a *sigmoid* when $C = 1$ and when we tolerate some small error.

Proposition 4.2. *Let the output activation be a sigmoid defined by* $g^{[2]}(u) = \frac{e^u}{1+e^u}$. *And let the encoding be linear as described in section 4.1. Then, we can find a solution* $W^{[2]}$, $\mathbf{b}^{[2]} \in \mathbb{R}^n$ *for the decoder, for which there exist an* $\epsilon \in \mathbb{R}^+$ *arbitrarily small such that*

$$MSE = \frac{1}{n(n+1)} \sum_{i=1}^{n+1} ||\mathbf{y}^{(i)} - \mathbf{x}^{(i)}||_2^2 < \epsilon \tag{11}$$

Proof. This proof consists in finding a solution $W^{[2]}$ and $\mathbf{b}^{[2]}$ for the decoder that makes $\mathbf{y}^{(i)}$ arbitrarily close to $\mathbf{x}^{(i)}$. Without loss of generality, let's suppose that $z^{(i)} = i$.

Looking at (2), for a fixed column j, we have

$$\mathbf{x}_j^{(i)} = 1 \quad \text{for } i = 1, ..., j \tag{12}$$

$$\mathbf{x}_j^{(i)} = 0 \quad \text{for } i = j + 1, ..., n + 1 \tag{13}$$

And writing again the equations (7) and (8) for an entry j and $z^{(i)} = i$,

$$\mathbf{y}_j^{(i)} = g^{[2]}(z^{(i)} W_j^{[2]} + \mathbf{b}_j^{[2]}) = \frac{1}{1 + e^{-(iW_j^{[2]} + \mathbf{b}_j^{[2]})}} \tag{14}$$

Consequently and first, we should have $W_j^{[2]} < 0$. Second, the inflection point of the *sigmoid* should occur between $z^{(j)} = j$ and $z^{(j+1)} = j + 1$. Recall that the inflection point occurs at 0. Thus, $W^{[2]}$ and $\mathbf{b}^{[2]}$ should be such that,

$$jW_j^{[2]} + \mathbf{b}_j^{[2]} > 0 \tag{15}$$

$$(j + 1)W_j^{[2]} + \mathbf{b}_j^{[2]} < 0 \tag{16}$$

As a result,

$$-(j + 1) < \frac{\mathbf{b}_j^{[2]}}{W_j^{[2]}} < -j \tag{17}$$

We will fix $\frac{\mathbf{b}_j^{[2]}}{W_j^{[2]}} = k$ with $k \in]-(j + 1), -j[$.

The above conditions ensure that for $i = 1, ..., j$, $y_j^{(i)} > 0.5$. However, for this range of i we wish $y_j^{(i)}$ to be as close as possible to 1. The largest error would be on $i = j$ as it is the closest to the inflection point (transition) of the *sigmoid*. The error on $i = j$ is

$$e^1 = (\mathbf{y}_j^{(j)} - 1)^2 = \left(\frac{1}{1 + e^{-(jW_j^{[2]} + b_j^{[2]})}} - 1 \right)^2 = \frac{1}{\left(1 + e^{W_j^{[2]}(j+k)}\right)^2} \tag{18}$$

Similarly, for $i = j + 1, ..., n + 1$ the largest error would be on $i = j + 1$ and it is equal to

$$e^0 = (\mathbf{y}_j^{(j+1)} - 0)^2 = \frac{1}{\left(1 + e^{-((j+1)W_j^{[2]} + b_j^{[2]})}\right)^2} = \frac{1}{\left(1 + e^{-W_j^{[2]}((j+1)+k)}\right)^2} \tag{19}$$

Note that the errors $e^1 = (\mathbf{y}_j^{(j)} - 1)^2 \underset{|W_j^{[2]}| \to \infty}{\longrightarrow} 0$ and $e^0 = (\mathbf{y}_j^{(j+1)} - 0)^2 \underset{|W_j^{[2]}| \to \infty}{\longrightarrow} 0$. This is logical since the *sigmoid* approaches a threshold when $|W_j^{[2]}| \to \infty$.

Now, we can find a solution having the following particularities. First, $W^{[2]}$ is uniform. That is, for all $j = 1, ..., n$, $W_j^{[2]} = w$ where w is a negative constant. Second, we take $k = -j - 0.5$. Thus, $\mathbf{b}_j^{[2]} = -wj - 0.5w$ will be affine with j. These two particularities lead to equal errors $e^0 = e^1$.

Subsequently, this solution has one parameter w that we can choose according to the desired MSE. Indeed, let the errors e^0 and e^1 be equal to a constant $e \in \mathbf{R}^+$. The total error is

$$\begin{aligned} MSE &= \frac{1}{n(n+1)} \sum_{i=1}^{n+1} \sum_{j=1}^{n} (\mathbf{y}_j^{(i)} - \mathbf{x}_j^{(i)})^2 \\ &< \frac{1}{n(n+1)} \sum_{j=1}^{n} je^1 + (n + 1 - j)e^0 \\ &< \frac{1}{n(n+1)} \sum_{j=1}^{n} je + (n + 1 - j)e = e \end{aligned}$$

Finally, we choose w to have $e = \epsilon << 1$. □

The experiments for this part are well detailed in this notebook.

4.3 Two Layers Decoder

In section 4.2, we saw that *ReLU* don't satisfy the necessary condition stated in lemma 4.1 to correctly reconstruct step functions with a one layer encoder and a scalar code. In this part, we will explore the reconstruction capacities of the decoder when we add a hidden layer to it and use only *ReLU* as a non-linearity.

Let's consider a two layer decoder as in fig.1. We detail the annotations and equations in these two layers.

- Layer 2:
  - Input: $\underset{(1\times 1)}{z^{(i)}}$
  - Output: $\underset{(1\times n^{[2]})}{\mathbf{a}^{[2](i)}}$
  - Parameters: $\underset{(n^{[2]}\times 1)}{W^{[2]}}$, $\underset{(1\times n^{[2]})}{\mathbf{b}^{[2]}}$
  - Equations:

$$\mathbf{u}^{[2](i)} = z^{(i)} {W^{[2]}}^T + \mathbf{b}^{[2]} \tag{20}$$

$$\mathbf{a}^{[2](i)} = g^{[2]}(\mathbf{u}^{[2](i)}) \tag{21}$$

- Layer 3:
  - Input: $\underset{(1\times n^{[2]})}{\mathbf{a}^{[2](i)}}$
  - Output: $\underset{(1\times n)}{\mathbf{y}^{(i)}}$
  - Parameters: $\underset{(n\times n)}{W^{[3]}}$
  - Equations:

$$\mathbf{u}^{[3](i)} = \mathbf{a}^{[2](i)} {W^{[3]}}^T \tag{22}$$

$$\mathbf{y}^{(i)} = g^{[3]}(\mathbf{u}^{[2](i)}) \tag{23}$$

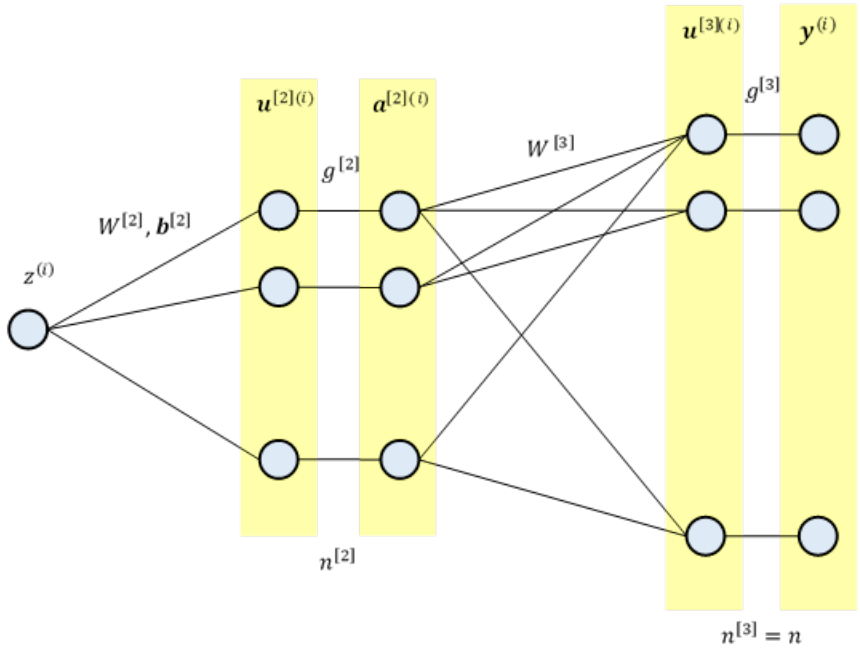

Figure 1: This is the decoder part of the autoencoder. The decoder is fed a latent code $z^{(i)}$ which is a scalar and outputs a vector $\mathbf{y}^{(i)}$ of dimension n. The layer 2 with $n^{[2]}$ neurons takes the latent code as input and outputs the activation vector $\mathbf{a}^{[2](i)}$.

Let's fix $g^{[3]}$ as a *ReLU*. What remains to set in this architecture is the number of neurons $n^{[2]}$ in the layer 2 and the activation $g^{[2]}$. We will do so based on the requirements of a perfect reconstruction.

Lemma 4.3. *Let M be a matrix of dimension $(n+1)\times n$ of the following form*

$$\underset{(n+1)\times n}{M} = \begin{bmatrix} C & C & \dots & C & C \\ m_{2,1} & C & \dots & C & C \\ m_{3,1} & m_{3,2} & \dots & C & C \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \dots & m_{n,n-1} & C \\ m_{n+1,1} & m_{n+1,2} & \dots & m_{n+1,n-1} & m_{n+1,n} \end{bmatrix}$$

where $m_{i,j} \leq 0\ \forall i,j$ such that $i = 2,...,n+1$ and $j = 1,...,i-1$. And C is a constant in $\mathbb{R}^+$.

Then M is a full row rank matrix.

Proof. Let $\mathbf{m}_i$ be a row of the matrix M. The proof consists in studying the linearly independence of the set of vectors $\{\mathbf{m}_i\}_{(i=1,...,n)}$ (no need to consider the last null row).

Let's find a set of scalars $\{\alpha_i\}_{(i=1,...,n)}$ such that

$$\sum_{i=1}^{n} \alpha_i \mathbf{m}_i = \mathbf{0} \tag{24}$$

All entries of the above combination should be null. Starting from entry $j = n$ and decreasing we'll have:

- for $j = n$:

$$\sum_{i=1}^{n} \alpha_i m_{i,n} = C \sum_{i=1}^{n} \alpha_i = 0 \tag{25}$$

- for $j = n-1$:

$$\sum_{i=1}^{n} \alpha_i m_{i,n-1} = C \sum_{i=1}^{n-1} \alpha_i + \alpha_n m_{n,n-1} = 0 \tag{26}$$

Then using (25) in (26) will lead to,

$$\alpha_n (m_{n,n-1} - C) = 0$$

This implies, $\alpha_n = 0$ or $m_{n,n-1} = C$. But $m_{n,n-1} \leq 0$ and $C > 0$, hence $\alpha_n = 0$. By recurrence, we can prove that $\forall i = 1,...,n$ we'll have $\alpha_i = 0$.

Consequently, (24) can only be satisfied if all $\alpha_i = 0$. Hence, the set $\{\mathbf{m}_i\}_{(i=1,...,n)}$ is linearly independent and M is full rank: $rank(M) = n$. □

Lemma 4.4. *Let $U^{[3]}$ be the preactivation matrix of the output layer where each row i is a preactivation $\mathbf{u}^{[3](i)}$.*

$$\underset{(n+1)\times n}{U^{[3]}} = \begin{bmatrix} \dots & \mathbf{u}^{[3](1)} & \dots \\ \dots & \mathbf{u}^{[3](2)} & \dots \\ \dots & \mathbf{u}^{[3](3)} & \dots \\ & \vdots & \\ \dots & \mathbf{u}^{[3](n)} & \dots \\ \dots & \mathbf{u}^{[3](n+1)} & \dots \end{bmatrix} \tag{27}$$

A perfect reconstruction $Y = X$ as in (4.1), is achieved only if $U^{[3]}$ is full rank matrix of the following form,

$$U^{[3]} = \begin{bmatrix} C & C & \dots & C & C \\ u_1^{[3](2)} & C & \dots & C & C \\ u_1^{[3](3)} & u_2^{[3](3)} & \dots & C & C \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ u_1^{[3](n)} & u_2^{[3](n)} & \dots & u_{n-1}^{[3](n)} & C \\ u_1^{[3](n+1)} & u_2^{[3](n+1)} & \dots & u_{n-1}^{[3](n+1)} & u_n^{[3](n+1)} \end{bmatrix} \tag{28}$$

where, $u_j^{[3](i)} \leq 0 \ \forall i, j$ *such that* $i = 2, ..., n+1$ *and* $j = 1, ..., i-1$.

Proof. Recall that by taking $g^{[3]} = ReLU$ we can rewrite (23) as

$$\mathbf{y}^{(i)} = ReLU(\mathbf{u}^{[3](i)}) = max(0, \mathbf{u}^{[3](i)}) \tag{29}$$

When the reconstruction is perfect, $\mathbf{y}_j^{(i)} \in \{0, C\}$ for all $i \in \{1, ..., n+1\}$ and $j \in \{1, ..., n\}$. Then, $\mathbf{y}_j^{(i)} = C > 0 \implies \mathbf{u}_j^{[3](i)} = \mathbf{y}_j^{(i)} = C$. Similarly, $\mathbf{y}_j^{(i)} = 0 \implies \mathbf{u}_j^{[3](i)} \leq 0$. This proves the particular form of $U^{[3]}$ in (28).

Moreover, using lemma 4.3 we obtain $rank(U^{[3]}) = n$ □

Now that we have a necessary condition on $U^{[3]}$, we will use it to deduce conditions on the previous layer and hence find the required $n^{[2]}$.

Lemma 4.5. *The preactivation matrix* $U^{[3]}$ *can be full rank under three necessary conditions:*

(i) The number of neurons $n^{[2]}$ *in the layer 2 is at least equal to* n

(ii) The weight matrix $W^{[3]}$ *is also be full rank*

(iii) Among the $n+1$ *generated activation vectors* $\{\mathbf{a}^{[2](i)}\}_{i=1,...,n+1}$*, at least* n *are linearly independent*

Proof. Let $A^{[2]}$ be the activation matrix of layer 2 whose rows are the the output of the second layer $\mathbf{a}^{[2](i)}$.

$$\underset{(n+1)\times n^{[2]}}{A^{[2]}} = \begin{bmatrix} \dots & \mathbf{a}^{[2](1)} & \dots \\ \dots & \mathbf{a}^{[2](2)} & \dots \\ \dots & \mathbf{a}^{[2](3)} & \dots \\ & \vdots & \\ \dots & \mathbf{a}^{[2](n)} & \dots \\ \dots & \mathbf{a}^{[2](n+1)} & \dots \end{bmatrix} \tag{30}$$

We write (22) in a matrix form

$$U^{[3]} = A^{[2]} {W^{[3]}}^T \tag{31}$$

And we apply the rank of a matrix product,

$$rank(U^{[3]}) \leq \min\left(rank(\underset{(n+1)\times n^{[2]}}{A^{[2]}}), rank(\underset{n\times n^{[2]}}{W^{[3]}})\right)$$

Hence, $rank(U^{[3]}) = n$ requires first to set $n^{[2]}$ to at least n. Second, $W^{[3]}$ should be full rank. And third, we must have $rank(A^{[2]}) \geq n$. This latter is equivalent to the condition (iii). □

Following lemma (4.5), we fix the number of the added hidden layer to n. Since the weights $W^{[3]}$ are learnt, we will suppose that having all other conditions satisfied, the autoencoder can manage to learn a full rank $W^{[3]}$. The remaining condition to satisfy is a full rank $A^{[2]}$.

Lemma 4.6. *The decoder can learn to generate* $A^{[2]}$ *with full rank if and only if the activation is a non-linearity and in the presence of a bias.*

Proof. The first part of the prove consists in demonstrating that a non-linearity and a bias are necessary. And this demonstration is a proof by contradiction.

Let's recall how $A^{[2]}$ is computed.

$$A^{[2]} = \begin{bmatrix} \dots & \mathbf{a}^{[2](1)} & \dots \\ \dots & \mathbf{a}^{[2](2)} & \dots \\ & \vdots & \\ \dots & \mathbf{a}^{[2](n+1)} & \dots \end{bmatrix} = \begin{bmatrix} \dots & \mathbf{g}^{[2]}(z^{(1)}{W^{[2]}}^T + \mathbf{b}^{[2]}) & \dots \\ \dots & \mathbf{g}^{[2]}(z^{(2)}{W^{[2]}}^T + \mathbf{b}^{[2]}) & \dots \\ & \vdots & \\ \dots & \mathbf{g}^{[2]}(z^{(n+1)}{W^{[2]}}^T + \mathbf{b}^{[2]}) & \dots \end{bmatrix} \tag{32}$$

If the activation function is linear, that is $g^{[2]}(u) = u$, then

$$A^{[2]} = \begin{bmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(n+1)} \end{bmatrix} {W^{[2]}}^T + \begin{bmatrix} \dots & \mathbf{b}^{[2]} & \dots \\ \dots & \mathbf{b}^{[2]} & \dots \\ & \vdots & \\ \dots & \mathbf{b}^{[2]} & \dots \end{bmatrix}$$

The two terms are obviously rank 1 matrices. Using the rank of a matrix sum,

$$rank(A^{[2]}) \leq rank\left(\begin{bmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(n+1)} \end{bmatrix} {W^{[2]}}^T\right) + rank\left(\begin{bmatrix} \dots & \mathbf{b}^{[2]} & \dots \\ \dots & \mathbf{b}^{[2]} & \dots \\ & \vdots & \\ \dots & \mathbf{b}^{[2]} & \dots \end{bmatrix}\right)$$

$$rank(A^{[2]}) \leq 2$$

As a consequence, if $A^{[2]}$ is full rank then the activation is non-linear in layer 2.

Furthermore, since we favour a *ReLU* activation function, a bias is also needed in order to achieve a full rank matrix. Indeed, having $g^{[2]} = ReLU$, without a bias, we'll have

$$g^{[2]}(z^{(i)}{W^{[2]}}^T) = \begin{cases} |z^{(i)}| g^{[2]}({W^{[2]}}^T) & \text{if } z^{(i)} \geq 0 \\ |z^{(i)}| g^{[2]}(-{W^{[2]}}^T) & \text{if } z^{(i)} < 0 \end{cases} \tag{33}$$

Thus, without a bias and with a ReLU activation function, the matrix $A^{[2]}$ will have at most a rank equal to 2. Subsequently, $A^{[2]}$ can be full rank only if layer 2 has both a non-linearity and a bias.

In the second part, we will prove that this condition is sufficient by finding a particular set of solutions that will make $A^{[2]}$ full rank.

Let's rewrite $A^{[2]}$ while detailing each element.

$$A^{[2]} = \begin{bmatrix} g^{[2]}(W_1^{[2]}(z^{(1)} + k_1)) & g^{[2]}(W_2^{[2]}(z^{(1)} + k_2)) & \dots & g^{[2]}(W_n^{[2]}(z^{(1)} + k_n)) \\ g^{[2]}(W_1^{[2]}(z^{(2)} + k_1)) & g^{[2]}(W_2^{[2]}(z^{(2)} + k_2)) & \dots & g^{[2]}(W_n^{[2]}(z^{(2)} + k_n)) \\ \vdots & \vdots & & \vdots \\ g^{[2]}(W_1^{[2]}(z^{(n+1)} + k_1)) & g^{[2]}(W_2^{[2]}(z^{(n+1)} + k_2)) & \dots & g^{[2]}(W_n^{[2]}(z^{(n+1)} + k_n)) \end{bmatrix} \tag{34}$$

where $k_j = \frac{\mathbf{b}_j^{[2]}}{W_j^{[2]}}$ for $j \in \{1, ..., n\}$.

Without loss of generality, let $z^{[i]}$ be monotonically increasing with i. We can find a set of trivial solutions to make $A^{[2]}$ full rank. Let's take $W_j^{[2]} < 0$ for all $j = 1, ..., n$. Then, we can choose

$$k_j = -\frac{z^{(j)} + z^{(j+1)}}{2} \tag{35}$$

As a result, we'll obtain $A^{[2]}$ full rank having the following form,

$$A^{[2]} = \begin{bmatrix} W_1^{[2]}(z^{(1)} + k_1) & W_2^{[2]}(z^{(1)} + k_2) & \dots & W_n^{[2]}(z^{(1)} + k_n) \\ 0 & W_2^{[2]}(z^{(2)} + k_2) & \dots & W_n^{[2]}(z^{(2)} + k_n) \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & W_n^{[2]}(z^{(n)} + k_n) \\ 0 & 0 & \dots & 0 \end{bmatrix} \tag{36}$$

Note that $z^{(i)}$ needs not to be monotonically increasing with respect to neither i nor to the positions $\theta^{(i)}$. Rather, the only necessary condition on $z^{(i)}$ is its uniqueness with respect to the positions $\theta^{(i)}$. That is, the codes $z^{(i)}$ should necessarily be different one another to achieve a full rank $A^{[2]}$.

Indeed, if $z^{(i)}$ is not monotonically increasing with i, we can always find a permutation matrix P that will sort the rows in an increasing order of $z^{(i)}$. Then we can apply the above particular solutions. □

Proposition 4.7. *Let's drop the signal identically equal to* 0 *from the dataset. A two layers decoder can perfectly reconstruction step functions from their scalar codes if and only if the layer 2 satisfies the following conditions:*

(i) Layer 2 has at least $n^{[2]} = n$ *neurons, where* n *is the dimension of the signal.*

(ii) The activation function $g^{[2]}$ *of layer 2 is non-linear. In particular,* $g^{[2]} = ReLU$*.*

(iii) Layer 2 has a bias vector.

Proof. Let $n^{[2]} = n$. Lemma 4.6 sets the conditions (ii) and (iii) to generate $A^{[2]}$ full rank. Then for all $U^{[3]}$ having the form in (28), there exists a full rank matrix $W^{[3]}$ such that ${W^{[3]}}^T = {A^{[2]}}^{-1} U^{[3]}$. Such matrices $U^{[3]}$, $W^{[3]}$ and $A^{[2]}$ satisfy both lemmas (4.4) and (4.5). As a consequence, these three conditions are necessary and sufficient for a perfect reconstruction. □

Note that the conditions in proposition 4.7 are also necessary to reconstruct the $n+1$ step functions including the signal identically equal to 0. However, this particularity requires further conditions that are not studied in this paper.

Corollary 4.7.1. *Once the three conditions of proposition 4.7 are satisfied, building a solution for a perfect reconstruction is primarily based on finding* $W^{[2]}$ *and* $\mathbf{b}^{[2]}$ *that make* $A^{[2]}$ *full rank with respect to the codes* $z^{[i]}$*. Subsequently, the third parameter* $W^{[3]}$ *is computed by choosing any matrix* $U^{[3]}$ *of the form (28) as such*

$$W^{[3]} = \left({A^{[2]}}^{-1} U^{[3]} \right)^T$$

4.4 Set of solutions

Let's consider a decoder satisfying the conditions in 4.7. And let the encoder represent a bijection. Such that for each unique position $\theta^{(i)}$, the encoder generates a unique code $z^{(i)}$.

Lemma 4.8. *Let* $\{\mathbf{a}^{(i)}\}_{i=1,\dots,n+1}$ *be the set of rows of* $A^{[2]}$*. If* $A^{[2]}$ *is full rank, then at most two rows have zeros at identical positions.*

Proof. The proof is a deduction from the affine dependence between the preactivation points $\mathbf{u}^{[2](i)}$.

Recall that

$$\mathbf{a}^{(i)} = ReLU(\mathbf{u}^{(i)}) = ReLU(z^{(i)} {W^{[2]}}^T + \mathbf{b}^{[2]})$$

Then $\mathbf{a}^{(i)}$ can be written as

$$\mathbf{a}^{(i)} = \mathbf{u}^{[2](i)} * \mathbb{1}^{(i)} = z^{(i)} ({W^{[2]}}^T * \mathbb{1}^{(i)}) + (\mathbf{b}^{[2]} * \mathbb{1}^{(i)})$$

where $*$ is an item-wise product of vectors. The vector $\mathbb{1}^{(i)} \in \{0,1\}^n$ is such that for each entry $j = 1, ..., n$, $\mathbb{1}^{(i)}_j = 0$ if $\mathbf{a}^{(i)}_j = 0$ and $\mathbb{1}^{(i)}_j = 1$ otherwise. Moreover, we denote

$$\psi^T = {W^{[2]}}^T * \mathbb{1}^{(i)} \quad \text{and} \quad \rho = \mathbf{b}^{[2]} * \mathbb{1}^{(i)}$$

Let's consider three samples of indices i, i' and $i'' \in \{1, ..., n+1\}$ corresponding respectively to $(z^{(i)}, \mathbf{a}^{(i)})$, $(z^{(i')}, \mathbf{a}^{(i')})$ and $(z^{(i'')}, \mathbf{a}^{(i'')})$. If these three samples have identical positions of 0, then $\mathbb{1}^{(i)} = \mathbb{1}^{(i')} = \mathbb{1}^{(i'')}$. Subsequently,

$$\mathbf{a}^{(i)[2]} = z^{(i)}\psi^T + \rho, \quad \mathbf{a}^{(i')[2]} = z^{(i')}\psi^T + \rho \quad \text{and} \quad \mathbf{a}^{(i'')[2]} = z^{(i'')}\psi^T + \rho$$

Because of this affine dependence, we can always find $\gamma^{(i)}$, $\gamma^{(i')}$ constants in $\mathbb{R}$,

$$\gamma^{(i)} = \frac{z^{(i'')} - z^{(i')}}{z^{(i)} - z^{(i')}} \quad \text{and} \quad \gamma^{(i')} = -\frac{z^{(i'')} - z^{(i)}}{z^{(i)} - z^{(i')}}$$

Such that,

$$\mathbf{a}^{[2](i'')} = \gamma^{(i)}\mathbf{a}^{[2](i)} + \gamma^{(i')}\mathbf{a}^{[2](i')}$$

Hence, three rows of $A^{[2]}$ having identical positions of zero are linearly dependent. □

Lemma 4.9. *Let $\{\mathbf{a}^{[2]}_j\}_{j=1,...,n}$ be the set of columns of $A^{[2]}$.If $A^{[2]}$ is full rank, then at most two rows have zeros at identical positions.*

Proof. The proof is similar to that of lemma 4.8. □

Corollary 4.9.1. *Without loss of generality, let $z^{(i)}$ be monotonically increasing with respect to i. Let's write $A^{[2]}$ in the following form,*

$$A^{[2]} = \begin{bmatrix} g^{[2]} \begin{bmatrix} W^{[2]}_1(z^{(1)} + k_1) \\ W^{[2]}_1(z^{(2)} + k_1) \\ \vdots \\ W^{[2]}_1(z^{(n+1)} + k_1) \end{bmatrix} g^{[2]} \begin{bmatrix} W^{[2]}_2(z^{(1)} + k_2) \\ W^{[2]}_2(z^{(2)} + k_2) \\ \vdots \\ W^{[2]}_2(z^{(n+1)} + k_2) \end{bmatrix} \dots g^{[2]} \begin{bmatrix} W^{[2]}_n(z^{(1)} + k_n) \\ W^{[2]}_n(z^{(2)} + k_n) \\ \vdots \\ W^{[2]}_n(z^{(n+1)} + k_n) \end{bmatrix} \end{bmatrix} \tag{37}$$

where

$$\mathbf{k} = [k_1, k_2, ..., k_n] = [\frac{\mathbf{b}^{[2]}_1}{W^{[2]}_1}, \frac{\mathbf{b}^{[2]}_2}{W^{[2]}_2}, ..., \frac{\mathbf{b}^{[2]}_n}{W^{[2]}_n}]^T \tag{38}$$

The positions of the zeros of $A^{[2]}$ are defined by $\mathbf{k}$ and the signs in $W^{[2]}$. Thus, the rank of $A^{[2]}$ only depends on $\mathbf{k}$ and the signs in $W^{[2]}$.

Proposition 4.10. *Let $\mathcal{U}$ be the set of all $U^{[3]}$ having the form (28). And let $\mathcal{K}$ be the set of all $\mathbf{k}$ that make the activation matrix of layer 2 full rank.*

For each fixed couple $\left(\mathbf{k}, U^{[3]}\right) \in \mathcal{K} \times \mathcal{U}$, there exist a set of infinitely many solutions described by the two following equations,

$$W^{[2]} = {W^{[3]}}^{-1} {W^{[2]}}^* \tag{39}$$

$$\mathbf{b}^{[2]} = {\mathbf{b}^{[2]}}^* {W^{[3]}}^{-1^T} \tag{40}$$

where ${W^{[2]}}^*$ is a constant matrix of dimension $n \times 1$ and ${b^{[2]}}^*$ is a constant vector of dimension n.

Proof. The equation of layer 3

$$U^{[3]} = A^{[2]} {W^{[3]}}^T \tag{41}$$

implies that for any $A^{[2]}$ full rank, we can always find $W^{[3]}$ full rank to obtain any $U^{[3]} \in \mathcal{U}$. Thus $A^{[2]}$ and $U^{[3]}$ can be chosen independently to form a solution for the perfect reconstruction.

Moreover, by definition, if $\mathbf{k} \in \mathcal{K}$, then there exists ${W^{[2]}}^+$ and ${b^{[2]}}^+$ such that their corresponding ${A^{[2]}}^+$ is full rank.

Subsequently, for each couple $\left(\mathbf{k}, U^{[3]}\right) \in \mathcal{K} \times \mathcal{U}$ there exists at least one solution for a perfect reconstruction ${W^{[2]}}^+$, ${\mathbf{b}^{[2]}}^+$ and ${W^{[3]}}^+ = \left({A^{[2]}}^{+^{-1}} U^{[3]}\right)^T$.

Let $D = diag(\alpha_1, \alpha_2, ..., \alpha_n)$ be a diagonal matrix, where $\alpha_j > 0$ for all $j = 1, ..., n$. And let

$$W^{[2]} = D{W^{[2]}}^+ \tag{42}$$

$$\mathbf{b}^{[2]} = {\mathbf{b}^{[2]}}^+ D \tag{43}$$

Then the couples, $(W^{[2]}, \mathbf{b}^{[2]})$ and $({W^{[2]}}^+, {\mathbf{b}^{[2]}}^+)$ correspond to the same $\mathbf{k}$. Moreover, because all $\alpha_j > 0$,

$$\begin{aligned} A^{[2]} &= ReLU(Z{W^{[2]}}^T + \mathbf{b}^{[2]}) \\ &= ReLU(Z(D{W^{[2]}}^+)^T + {\mathbf{b}^{[2]}}^+ D) \\ &= ReLU(Z{W^{[2]}}^{+^T} + {\mathbf{b}^{[2]}}^+)D = {A^{[2]}}^+ D \end{aligned}$$

As a consequence $A^{[2]}$ is also full rank. Then for the same $U^{[3]}$, we can find a solution

$$\begin{aligned} W^{[3]} &= \left({A^{[2]}}^{-1} U^{[3]}\right)^T \\ &= \left(({A^{[2]}}^+ D)^{-1} U^{[3]}\right)^T = {W^{[3]}}^+ D^{-1} \end{aligned}$$

Hence, the triplet $(W^{[2]}, \mathbf{b}^{[2]}, W^{[3]})$ is a solution to a perfect reconstruction. Since $W^{[3]}$ is full rank, we can write,

$$D = {W^{[3]}}^{-1} {W^{[3]}}^+ \tag{44}$$

Finally, we obtain,

$$\begin{aligned} W^{[2]} &= D{W^{[2]}}^+ = {W^{[3]}}^{-1} {W^{[3]}}^+ {W^{[2]}}^+ = {W^{[3]}}^{-1} {W^{[2]}}^* \\ \mathbf{b}^{[2]} &= {\mathbf{b}^{[2]}}^+ D^T = {\mathbf{b}^{[2]}}^+ {W^{[3]}}^{+^T} {W^{[3]}}^{-1^T} = {\mathbf{b}^{[2]}}^* {W^{[3]}}^{-1^T} \end{aligned}$$

where,

$$\begin{aligned} {\mathbf{b}^{[2]}}^* &= {\mathbf{b}^{[2]}}^+ {W^{[3]}}^{+^T} \\ {W^{[2]}}^* &= {W^{[3]}}^+ {W^{[2]}}^+ \end{aligned}$$

□

4.5 Experiments

The experiments we conducted have mainly two purposes. On one hand, we want to verify the theoretical results. And on the other hand, we want gain some understanding regarding the optimization of the autoencoder. Therefore, in what follows, the architecture of the decoder will follow the conditions stated in proposition 4.7. Additionally, we will set $n = 128$ and $C = 5$.

The following experiments are found this notebook .

4.5.1 Handcrafted Solution with Linear Encoder

We will handcraft solutions according to corollary 4.7.1. This will help verify the theoretical parts concerning the perfect reconstruction ability of a two layers decoder.

First and for simplicity, we will consider a linear encoder that outputs $z^{(i)} = \theta^{(i)} = i$.

$$W^{[1]} = -\frac{1}{5}[1, 1, ..., 1] \quad \text{and} \quad b^{[1]} = n + 1 = 129$$

According to the theoretical results, for a single encoding function, there exists infinitely many solutions $(W^{[2]}, \mathbf{b}^{[2]}, W^{[3]})$. We will chose a simple full rank $A^{[2]}$ of the form (36) as in the proof of lemma 4.6.

$$k_j = \frac{\mathbf{b}_j^{[2]}}{W_j^{[2]}} = -j - 0.5, \quad W^{[2]} = [-1, -1, ..., -1] \quad \text{and} \quad \mathbf{b}^{[2]} = [1.5, 2.5, ..., 128.5]$$

The resulting $A^{[2]}$ is

$$A^{[2]} = \begin{bmatrix} 0.5 & 1.5 & \dots & 127.5 \\ 0 & 0.5 & \dots & 126.5 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 0.5 \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

To compute $W^{[3]}$, we arbitrarily choose $U^{[3]}$ in the form (28) with $C = 5$,

$$U^{[3]} \begin{bmatrix} 5 & 5 & \dots & 5 \\ -3 & 5 & \dots & 5 \\ \vdots & \vdots & & \vdots \\ -3 & -3 & \dots & 5 \\ -3 & -3 & \dots & -3 \end{bmatrix}$$

Finally, we compute $W^{[3]} = \left(A^{[2]^{-1}} U^{[3]}\right)^T$

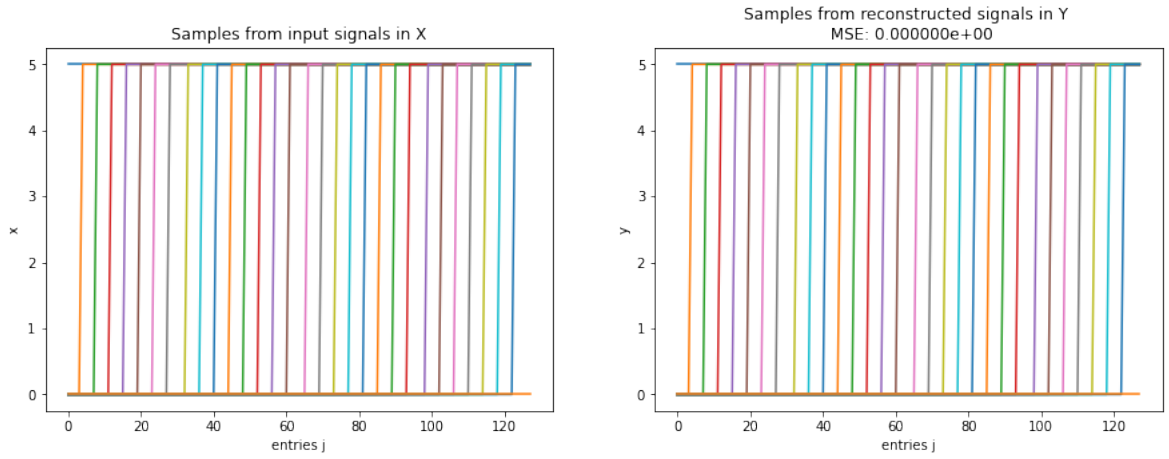

Figure 2: Result of handcrafted solution with linear encoder. On the left are the true input signal. On the right are the reconstructed ones. The MSE is null.

As we can see the reconstruction is perfect. This proves the existence of a solution with a linear encoding. Note that the signal identically equal to 0 is also perfectly reconstructed.

Next, in the proof of proposition 4.10, we mentioned that, for a fixed encoder, if $(W^{[2]}, \mathbf{b}^{[2]}, W^{[3]})$ is a solution, then for any positive diagonal matrix D, $(DW^{[2]}, \mathbf{b}^{[2]}D, W^{[3]}D^{-1})$ is also a solution. To test the validity of this statement, we will generate a random positive matrix D, compute the new parameters and then compute the MSE on the corresponding reconstructions.

Indeed, as we see in fig.3, the new set of parameters is also a solution. However, the resulting MSE is not exactly 0. Instead, it is in the order of 10^{-8} depending on the generated D. We believe this is due to numerical limitations.

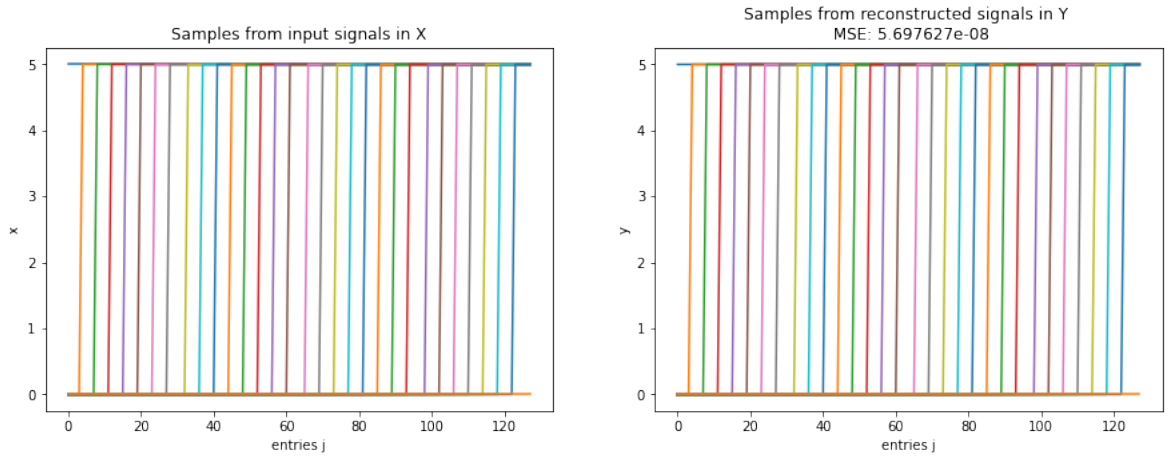

Figure 3: Result of a new generated solution. On the left are the true input signal. On the right are the reconstructed ones. The MSE is very small.

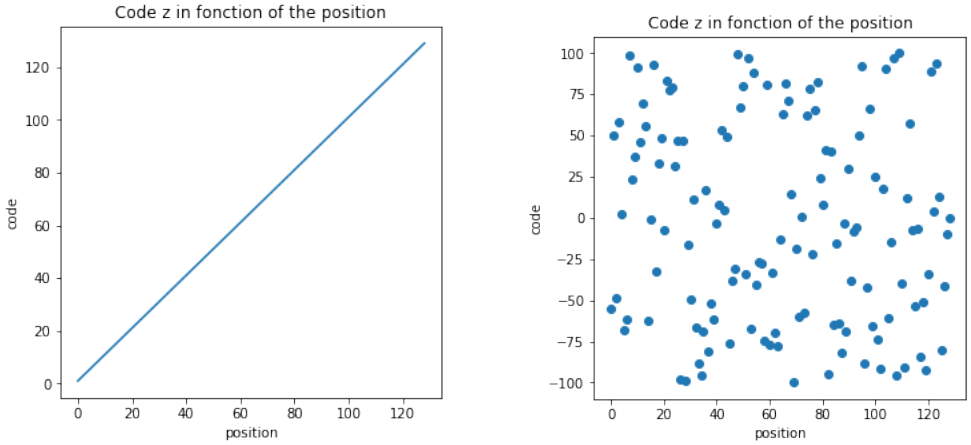

Figure 4: On the left is the linear code generated from the solution in section 4.5.1. On the right is the random code sampled in section 4.5.2.

4.5.2 Handcrafted Solution with non monotonic Encoder

In the proof of lemma 4.6, we noted that the only requirement on the codes is their uniqueness with respect to the positions $\theta^{(i)}$. We will prove the validity of this statement by building a solution with a non monotonic encoding.

The process of finding a solution is not detailed in this paper. But in a nutshell, whatever was the encoding, we can always sort the codes $z^{(i)}$ in an ascending order. And then find the parameters of Layer 2 using (35). The random codes are shown in fig.4. And the results are shown in fig.5.

Note that in this case, the decoder fails to reconstruct the signal identically equal to 0. For that, we omitted it from the set of signals to be shown and from the computation of the MSE.

4.6 Training with Linear Encoder

We constraint the encoder to be linear by fixing the weights of layer 1 as follows

$$W^{[1]} = w[1, 1, ..., 1]$$

where the constant $w \in \mathbb{R}$ is learnt. The bias $b^{[1]}$ is also learnt. This makes it easier for the autoencoder to find a solution.

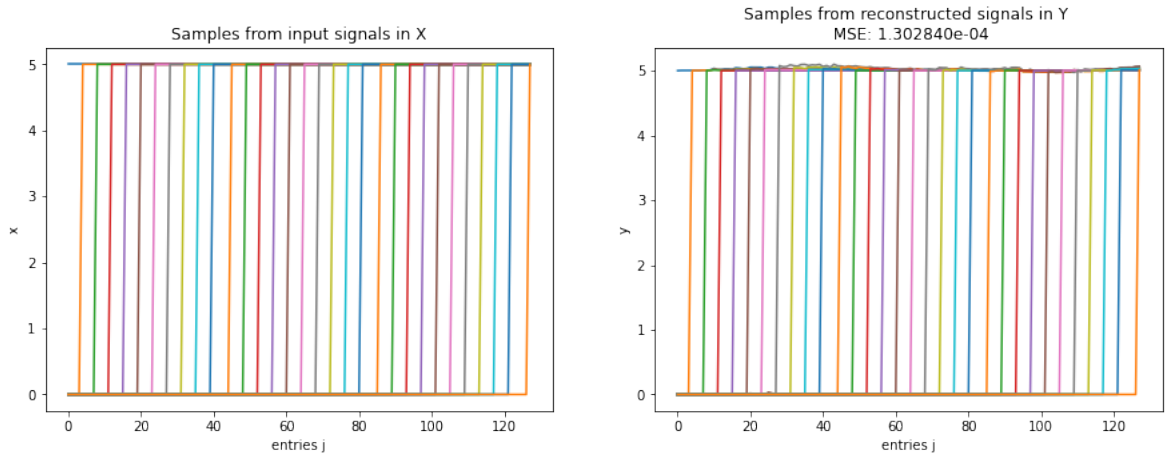

Figure 5: Result of the solution found for a random non monotonic encoder. On the left are the true input signal. On the right are the reconstructed ones. The MSE is very small.

We initialize the weights of layer 2 and layer 3 randomly following a uniform sampling on $[0, 0.1]$. This sampling is chosen to prevent dead neurons in the output layer. Moreover, we use the RMSprop optimizer as it makes the learning faster.

The results are depicted in fig.6. Although it is not perfect, we can say that the autoencoder was able to find a good solution. And this solution can get better if we let the autoencoder learn longer.

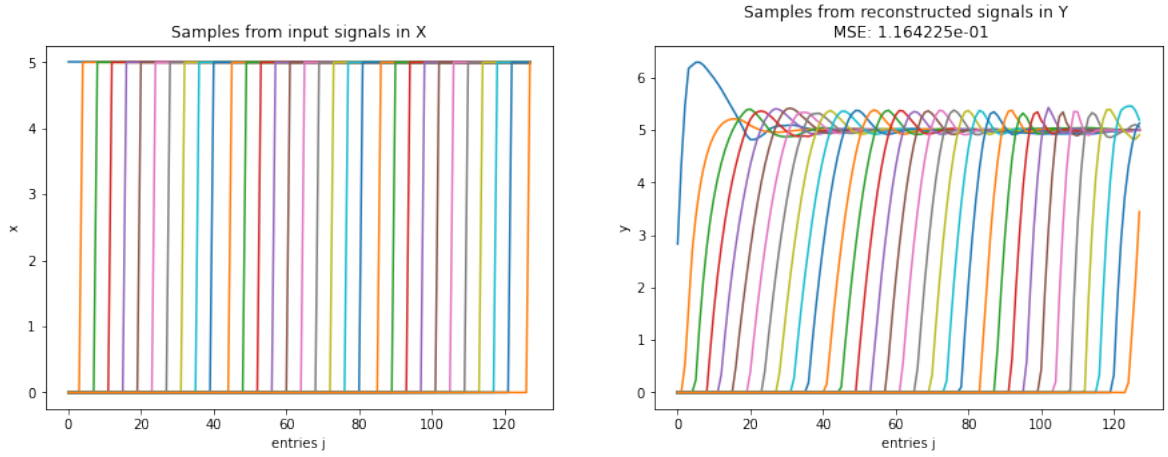

Figure 6: Results for training the autoencoder with a linear constraint on the encoder. On the left are the true input signal. On the right are the reconstructed ones.

However, the big problem in the optimization is that it is extremely slow. This result took a total of $2,500,000$ epochs. This is not acceptable in the case of 1D signals of dimension $n = 128$.

4.7 Training free Autoencoder

As a last experiment, we free the encoder and let all the weights be learnable. The initialization of the weights is done similarly to section 4.6. And we also use RMSprop. The results are shown in fig.7 and fig. 8.

We first notice that when freed, the encoder did not learn a linear representation of the positions. Second, the learnt code is monotonic with respect to the positions except for the last few samples.

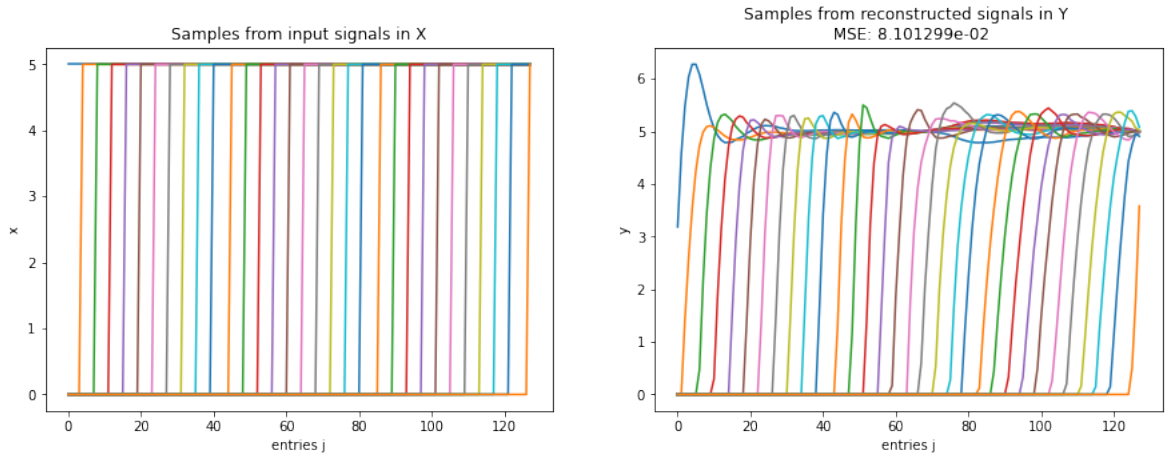

Figure 7: Results for training a free autoencoder. On the left are the true input signal. On the right are the reconstructed ones.

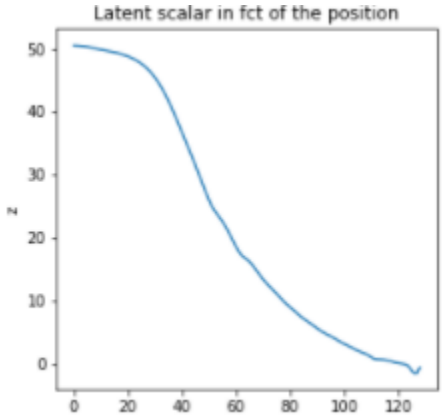

Figure 8: Plot of the codes versus the positions when the autoencoder was trained without constraints.

4.8 Conclusion

In this paper, we did a theoretical study to conceive the simplest autoencoder that can achieve a perfect reconstruction of step functions while having its encoder linear and a scalar code. Then we conducted some experiments to prove the theoretical results. As a conclusion, the autoencoder should have a one layer encoder and a two layers decoder. The first layer of the decoder is required to have a n neurons, a non-linearity and a bias. Then we saw that such architecture has infinitely many solutions. However, one of its main disadvantages was the slow training.