

# Autoencoders explained

Lorenzo Bozzoni

August 27, 2024

This small research is a theoretical project for the Numerical Analysis for Machine Learning course at Politecnico di Milano. In the following pages an introduction to the topic of autoencoders is presented including linear and boolean class and a final part on the complexity.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Linear Autoencoders</b>	<b>3</b>
2.1	Useful mathematical concepts . . . . .	3
2.2	Fact 1 . . . . .	4
2.2.1	Proof of fact 1 . . . . .	5
2.3	Fact 2 . . . . .	5
2.4	Fact 3 . . . . .	6
2.4.1	Proof of fact 3 . . . . .	6
2.5	Fact 4 . . . . .	7
2.5.1	Proof of fact 4 . . . . .	8
2.6	Equivalence with PCA . . . . .	10
<b>3</b>	<b>Boolean Autoencoders</b>	<b>11</b>
<b>4</b>	<b>Clustering complexity on the hypercube</b>	<b>14</b>
4.1	Brief recap on computational complexity . . . . .	14
4.2	Clustering complexity . . . . .	15
<b>5</b>	<b>Overcomplete Autoencoders</b>	<b>16</b>

## 1 Introduction

Autoencoders are simple learning circuits which aim to transform inputs into outputs with the least possible amount of distortion. To derive a fairly general framework, an  $n/p/n$  autoencoder is defined by a t-uple  $n, p, m, \mathbb{F}, \mathbb{G}, \mathcal{A}, \mathcal{B}, \mathcal{X}, \mathcal{Y}, \Delta$ , where:

- $\mathbb{F}$  and  $\mathbb{G}$  are sets
- $n$  and  $p$  are positive integers

- $\mathcal{A}$  is a class of functions from  $\mathbb{G}^p$  to  $\mathbb{F}^n$
- $\mathcal{B}$  is a class of functions from  $\mathbb{F}^n$  to  $\mathbb{G}^p$
- $\mathcal{X} = x_1, \dots, x_m$  is a set of  $m$  (training) vectors in  $\mathbb{F}^n$ . When the external targets are present, we let  $\mathcal{Y} = y_1, \dots, y_m$  denote the corresponding set of  $m$  target vectors in  $\mathbb{F}^n$
- $\Delta$  is a dissimilarity or distortion function defined over  $\mathbb{F}^n$

For any  $A \in \mathcal{A}$  and  $B \in \mathcal{B}$ , the autoencoder transforms an input vector  $x \in \mathbb{F}^n$  into an output vector  $A \circ B(x) \in \mathbb{F}^n$ .

$$\mathcal{X} \ni x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \xrightarrow{B} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_p \end{bmatrix} \xrightarrow{A} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = y \in \mathcal{Y}$$

The corresponding **autoencoder problem** is to find  $A \in \mathcal{A}$  and  $B \in \mathcal{B}$  that minimize the overall distortion function:

$$\min E(A, B) = \min_{A, B} \sum_{t=1}^m E(A, B) = \min_{A, B} \sum_{t=1}^m \Delta(x_t, A \circ B(x_t)) \quad (1)$$

In the non auto-associative case, when external targets  $y_t$  are provided, the minimization problem becomes:

$$\min E(A, B) = \min_{A, B} \sum_{t=1}^m E(A, B) = \min_{A, B} \sum_{t=1}^m \Delta(y_t, A \circ B(x_t)) \quad (2)$$

It is important to notice that if  $p < n$  corresponds to a compression or feature extraction, while  $p > n$  corresponds to a decompression. Here is a simple classification of the different types of autoencoders:

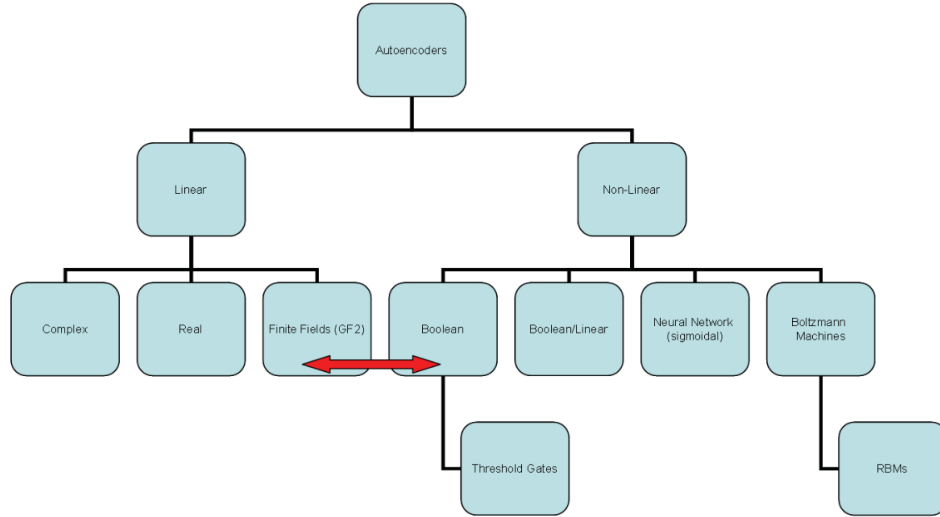


Figure 1: Simple classification of autoencoders

We are going to explore linear real and boolean autoencoders in the following sections.

## 2 Linear Autoencoders

We consider the problem of learning from examples in layered linear feed-forward neural networks using optimization methods, such as back propagation, with respect to the usual quadratic error function  $E$  of the connection weights.

We assume to have  $N$  samples and  $N$  labels so for each  $x_n$  input vector corresponds the  $y_n$  label. The classical quadratic error function is defined as:

$$E = \sum_{t=1}^m \|y_t - F(x_t)\|^2$$

where  $F$  is the current function implemented by the network. We defined also the **covariance matrices**:

$$\begin{aligned} \Sigma_{XX} &= \sum_{t=1}^m x_t x_t^\top & \Sigma_{YX} &= \sum_{t=1}^m y_t x_t^\top \\ \Sigma_{YY} &= \sum_{t=1}^m y_t y_t^\top & \Sigma_{XY} &= \sum_{t=1}^m x_t y_t^\top \end{aligned}$$

Where these quantities are defined.

Of course the aim is not trivially to minimize the error function, since this would be simply solved by choosing as global mapping  $W = AB = I$ , but rather to find  $W$  such that, having a smaller rank, it performs a good compression of the input data.

### 2.1 Useful mathematical concepts

For any matrices  $P, Q, R$  we have  $tr(PQR) = tr(RPQ) = tr(QRP)$  provided that these quantities are defined. Thus in particular if  $P$  is **idempotent**, that is,  $P^2 = P$ , then:

$$tr(PQP) = tr(PPQ) = tr(P^2Q) = tr(PQ) \quad (a)$$

If  $U$  is orthogonal, that is  $U^\top U = I$ , then:

$$tr(UQU^\top) = tr(U^\top UQ) = tr(Q) \quad (b)$$

The **Kronecker product**  $P \otimes Q$  of any two matrices  $P$  and  $Q$  is the matrix obtained from the matrix  $P$  by replacing each entry  $p_{ij}$  of  $P$  with the matrix  $p_{ij}Q$ . Which means that:

$$P : m \times n \text{ and } Q : r \times q \implies P \otimes Q = \begin{bmatrix} p_{11}Q & \dots & p_{1n}Q \\ \vdots & \ddots & \vdots \\ p_{m1}Q & \dots & p_{mn}Q \end{bmatrix} \text{ of shape } rm \times qn$$

The **vec operation** transforms a matrix into a column vector by stacking the columns of the matrix one underneath the other. Indeed, if  $P$  is any  $m \times n$  matrix and  $p_j$  is the  $j$ -th column, then  $vec(P)$  is the  $mn \times 1$  vector  $vec(P) = [p_1^\top, \dots, p_n^\top]^\top$ .

We have that:

$$\text{tr}(PQ^\top) = (\text{vec}(P))^\top \text{vec}(Q) \quad (\text{c})$$

$$\text{vec}(PQR^\top) = (R \otimes P) \text{vec}(Q) \quad (\text{d})$$

$$(P \otimes Q)(R \otimes S) = PR \otimes QS \quad (\text{e})$$

$$(P \otimes Q)^{-1} = P^{-1} \otimes Q^{-1} \quad (\text{f})$$

$$(P \otimes Q)^\top = P^\top \otimes Q^\top \quad (\text{g})$$

whenever these quantities are defined. Also: if  $P$  and  $Q$  are symmetric and positive semidefinite (resp. definite) then  $P \otimes Q$  is symmetric and positive semidefinite (resp. positive definite) (h).

Finally, let us introduce the input data matrix  $X = [x_1, \dots, x_N]$  and the output data matrix  $Y = [y_1, \dots, y_N]$ . It is easily seen that  $XX^\top = \Sigma_{XX}$ ,  $XY^\top = \Sigma_{XY}$ ,  $YY^\top = \Sigma_{YY}$ ,  $YX^\top = \Sigma_{YX}$  and  $E(A, B) = \|\text{vec}(Y - ABX)\|^2$ . In the proof of facts 1 and 2, we shall use the following well known lemma.

**Lemma:** the quadratic function:

$$F(z) = \|c - Mz\|^2 = c^\top c - 2c^\top Mz + z^\top M^\top Mz$$

is convex. A point  $z$  corresponds to a global minimum of  $F$  if and only if it satisfies the equation  $\nabla F = 0$ , or equivalently  $M^\top Mz = M^\top c$ . If in addition  $M^\top M$  is positive definite, then  $F$  is strictly convex and the unique minimum of  $F$  is attained for  $z = (M^\top M)^{-1} M^\top c$ .

## 2.2 Fact 1

For any fixed  $n \times p$  matrix  $A$  the function  $E(A, B)$  is convex in the coefficients of  $B$  and attains its minimum for any  $B$  satisfying the equation

$$A^\top AB \Sigma_{XX} = A^\top \Sigma_{YX} \quad (3)$$

If  $\Sigma_{XX}$  is invertible and  $A$  is full rank  $p$ , then  $E$  is strictly convex and has unique minimum reached when:

$$B = \hat{B}(A) = (A^\top A)^{-1} A^\top \Sigma_{YX} \Sigma_{XX}^{-1} \quad (3)$$

In the auto-associative case, (3) becomes

$$B = \hat{B}(A) = (A^\top A)^{-1} A^\top \quad (3')$$

Since  $Y = X$  so  $\Sigma_{YX} \Sigma_{XX}^{-1} = I$ .

### 2.2.1 Proof of fact 1

*Proof.* For fixed  $A$ , use (d) to write:

$$\text{vec}(Y - ABX) = \text{vec}(Y) - \text{vec}(ABX) = \text{vec}(Y) - (X^\top \otimes A)\text{vec}(B)$$

and thus:

$$E(A, B) = \|\text{vec}(Y) - (X^\top \otimes A)\text{vec}(B)\|^2$$

By the above lemma,  $E$  is convex in the coefficients of  $B$  and  $B$  corresponds to a global minimum if and only if

$$(X^\top \otimes A)^\top (X^\top \otimes A)\text{vec}(B) = (X^\top \otimes A)\text{vec}(Y)$$

Now on one hand:

$$\begin{aligned} (X^\top \otimes A)^\top (X^\top \otimes A)\text{vec}(B) &= (X^\top \otimes A)\text{vec}(B) \\ &= (XX^\top \otimes A^\top A)\text{vec}(B) \\ &= (\Sigma_{XX} \otimes A^\top A)\text{vec}(B) \\ &= \text{vec}(A^\top AB\Sigma_{XX}) \end{aligned}$$

On the other hand:

$$\begin{aligned} (X^\top \otimes A)^\top \text{vec}(Y) &= (X \otimes A^\top)\text{vec}(Y) \\ &= \text{vec}(A^\top YX^\top) \\ &= \text{vec}(A^\top \Sigma_{YX}) \end{aligned}$$

Therefore:

$$A^\top AB\Sigma_{XX} = A^\top \Sigma_{YX}$$

which is (2). If  $A$  is full rank,  $A^\top A$  is symmetric and positive definite. As a covariance matrix,  $\Sigma_{XX}$  is symmetric and positive semidefinite; if, in addition,  $\Sigma_{XX}$  is invertible, then  $\Sigma_{XX}$  is also positive definite. Because of (h),  $(X^\top \otimes A)^\top (X^\top \otimes A) = \Sigma_{XX} \otimes A^\top A$  is also symmetric and positive definite. Applying the above lemma, we conclude that if  $\Sigma_{XX}$  is invertible and  $A$  is a fixed full rank matrix, then  $E$  is strictly convex in the coefficients of  $B$  and attains its unique minimum at the unique solution  $B = \hat{B}(A) = (A^\top A)^{-1} A^\top \Sigma_{YX} \Sigma_{XX}^{-1}$  of (2), which is (3). In the auto-associative case,  $x_n = y_n$ . Therefore  $\Sigma_{XX} = \Sigma_{YX} = \Sigma_{YY} = \Sigma_{XY}$  and the above expression simplifies to (3').  $\square$

## 2.3 Fact 2

For any fixed  $p \times n$  matrix  $B$  the function  $E(A, B)$  is convex in the coefficients of  $A$  and attains its minimum for any  $A$  satisfying the equation:

$$AB\Sigma_{XX}B^\top = \Sigma_{YX}B^\top \tag{4}$$

If  $\Sigma_{XX}$  is invertible and  $B$  is full rank  $p$ , then  $E$  is strictly convex and has a unique minimum reached when:

$$A = \hat{A}(B) = \Sigma_{YX}B^\top (B\Sigma_{XX}B^\top)^{-1} \tag{5}$$

In the auto-associative case, (5) becomes:

$$A = \hat{A}(B) = \Sigma_{XX}B^\top (B\Sigma_{XX}B^\top)^{-1} \tag{5'}$$

The proof is nearly identical to the proof of fact 1.

## 2.4 Fact 3

Assume that  $\Sigma_{XX}$  is invertible. If two matrices  $A$  and  $B$  define a critical point of  $E$  (i.e. a point where  $\frac{\partial E}{\partial a_{ij}} = \frac{\partial E}{\partial b_{ij}} = 0$ ) then the global map  $W = AB$  is of the form:

$$W = P_A \Sigma_{YX} \Sigma_{XX}^{-1} \quad (6)$$

with  $A$  satisfying

$$P_A \Sigma = P_A \Sigma P_A = \Sigma P_A \quad (7)$$

Where  $\Sigma = \Sigma_{YX} \Sigma_{XX}^{-1} \Sigma_{XY}$ . Recall also, that the matrix  $P_A$  is the matrix of the orthogonal projection onto the subspace spanned by the columns of  $A$ . In the auto-associative case,  $\Sigma = \Sigma_{XX}$  and (6) and (7) become:

$$W = AB = P_A \quad (6')$$

$$P_A \Sigma_{XX} = P_A \Sigma_{XX} P_A = \Sigma_{XX} P_A \quad (7')$$

If  $A$  is full rank  $p$ , then  $A$  and  $B$  define a critical point of  $E$  if and only if  $A$  satisfies (7) and  $B = \hat{B}(A)$ , or equivalently if and only if  $A$  and  $W$  satisfy (6) and (7').

### 2.4.1 Proof of fact 3

*Proof.* Assume first that  $A$  and  $B$  define a critical point of  $E$ , with  $A$  full rank. Then from fact 1 we get  $B = \hat{B}(A)$  and thus

$$W = AB = A(A^\top A)^{-1} A \Sigma_{YX} \Sigma_{XX}^{-1} = P_A \Sigma_{YX} \Sigma_{XX}^{-1}$$

Which is (6). Multiplication of (4) by  $A^\top$  on the right yields

$$W \Sigma_{XX} W^\top = AB \Sigma_{XX} B^\top A^\top = \Sigma_{YX} B^\top A^\top = \Sigma_{YX} W^\top$$

Or

$$P_A \Sigma_{YX} \Sigma_{XX}^{-1} \Sigma_{XX} \Sigma_{XX}^{-1} \Sigma_{XY} P_A = \Sigma_{YX} \Sigma_{XX}^{-1} \Sigma_{XY} P_A$$

Or, equivalently  $P_A \Sigma P_A = \Sigma P_A$ . Since both  $\Sigma$  and  $P_A$  are symmetric,  $P_A \Sigma P_A = \Sigma P_A$  is also symmetric and therefore  $\Sigma P_A = (\Sigma P_A)^\top = P_A^\top \Sigma^\top = P_A \Sigma$ , which is (7). Hence if  $A$  and  $B$  correspond to a critical point and  $A$  is full rank then (6) and (7) must hold and  $B = \hat{B}(A)$ .

Conversely, assume that  $A$  and  $W$  satisfy (6) and (7), with  $A$  full rank. Multiplying (6) by  $(A^\top A)^{-1} A^\top$  on the left yields

$$B = (A^\top A)^{-1} A \Sigma_{YX} \Sigma_{XX}^{-1} = \hat{B}(A)$$

and (2) is satisfied. From  $P_A \Sigma P_A = \Sigma P_A$  and using (6) we immediately get

$$AB \Sigma_{XX} B^\top A^\top = \Sigma_{YX} B^\top A^\top$$

and multiplication of both sides by  $A(A^\top A)^{-1}$  on the right yields

$$AB \Sigma_{XX} B^\top = \Sigma_{YX} B^\top$$

which is (4). Thus  $A$  and  $B$  satisfy (2) and (4) and therefore they define a critical point of  $E$ .  $\square$

## 2.5 Fact 4

Assume that  $\Sigma$  is full-rank with  $n$  distinct eigenvalues  $\lambda_1 > \dots > \lambda_n$ . If  $\mathcal{I} = i_1, \dots, i_p$  ( $1 \leq i_1 < \dots < i_p \leq n$ ) is any ordered  $p$ -index set, let  $U_{\mathcal{I}} = [u_{i_1}, \dots, u_{i_p}]$  denote the matrix formed by the orthonormal eigenvectors of  $\Sigma$  associated with the eigenvalues  $\lambda_{i_1}, \dots, \lambda_{i_p}$ . Then two full rank matrices  $A$  and  $B$  define a critical point of  $E$  if and only if there exist an ordered  $p$ -index set  $\mathcal{I}$  and an invertible  $p \times p$  matrix  $C$  such that:

$$A = U_{\mathcal{I}} C \quad (8)$$

$$B = C^{-1} U_{\mathcal{I}}^T \Sigma_{YX} \Sigma_{XX}^{-1} \quad (9)$$

For such a critical point we have:

$$W = P_{U_{\mathcal{I}}} \Sigma_{YX} \Sigma_{XX}^{-1} \quad (10)$$

$$E(A, B) = \text{tr}(\Sigma_{YY}) - \sum_{i \in \mathcal{I}} \lambda_i \quad (11)$$

Therefore a critical  $W$  of rank  $p$  is always the product of the ordinary least squares regression matrix followed by an orthogonal projection onto the subspace spanned by  $p$  eigenvectors of  $\Sigma$ . The critical map  $W$  associated with the index set  $\{1, 2, \dots, p\}$  is the unique local and global minimum of  $E$ . The remaining  $\binom{n}{p} - 1$   $p$ -index sets correspond to saddle points. All additional critical points defined by matrices  $A$  and  $B$  which are not full rank are also saddle points and can be characterized in terms of orthogonal projections onto subspaces spanned by  $q$  eigenvectors of  $\Sigma$  with  $q < p$  (see Figure 1).

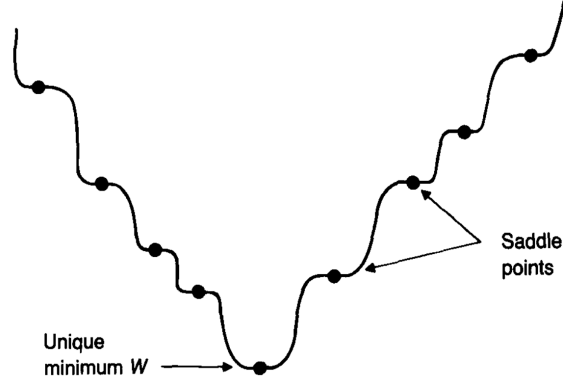


Figure 2: Landscape of  $E$

In the auto-associative case,  $\Sigma = \Sigma_{XX}$  and (8), (9) and (10) become:

$$A = U_{\mathcal{I}} C \quad (8')$$

$$B = C^{-1} U_{\mathcal{I}}^T \quad (9')$$

$$W = P_{U_{\mathcal{I}}} \quad (10')$$

and therefore the unique locally and globally optimal map  $W$  is the orthogonal projection onto the subspace spanned by the first  $p$  eigenvectors of  $\Sigma_{XX}$  associated with the  $p$  largest eigenvalues.

*Remark:* at the global minimum, if  $C$  is the identity  $I_p$  then the activities of the units in the hidden layer are given by:

$$u_1^\top \hat{y}_n, \dots, u_p^\top \hat{y}_n$$

the so called **principal components** of the output data  $\hat{y}$ . In the auto-associative case, these activities are given by  $u_1^\top x_n, \dots, u_p^\top x_n$ , the principal components of the input data  $x$ . They are the coordinates of the vector  $x_n$  along the first  $p$  eigenvectors of  $\Sigma_{XX}$ .

### 2.5.1 Proof of fact 4

First notice that since  $\Sigma$  is a real symmetric covariance matrix, it can always be written as  $\Sigma = U\Lambda U^\top$  where  $U$  is an orthogonal column matrix of eigenvectors of  $\Sigma$  and  $\Lambda$  is the diagonal matrix with non-increasing eigenvalues on its diagonal. Also if  $\Sigma$  is full-rank, then  $\Sigma_{XX}, \Sigma_{XY}, \Sigma_{YX}$  are full rank too. Now clearly if  $A$  and  $B$  satisfy (8) and (9) for some  $C$  and some  $\mathcal{I}$ , then  $A$  and  $B$  are full rank  $p$  and satisfy (3) and (5). Therefore they define a critical point of  $E$ .

For the converse, we have:

$$P_{U^\top A} = U^\top A(A^\top U U^\top A)^{-1} A^\top U = U^\top A(A^\top A)^{-1} A^\top U = U^\top P_A U$$

or, equivalently,  $P_A = U P_{U^\top A} U^\top$ . Hence (7) yields:

$$U P_{U^\top A} U^\top U \Lambda U^\top = P_A \Sigma = \Sigma P_A = U \Lambda U^\top U P_{U^\top A} U^\top$$

and so  $P_{U^\top A} \Lambda = \Lambda P_{U^\top A}$ . Since  $\lambda_1 > \dots > \lambda_n > 0$ , it is readily seen that  $P_{U^\top A}$  is an orthogonal projector of rank  $p$  and its eigenvalues are 1 ( $p$  times) and 0 ( $n - p$  times). Therefore there exists a unique index set  $\mathcal{I} = i_1, \dots, i_p$  with  $1 \leq i_1 < \dots < i_p \leq n$  such that  $P_{U^\top A} = I_{\mathcal{I}}$ , where  $I_{\mathcal{I}}$  is the diagonal matrix with entry  $i = 1$  if  $i \in \mathcal{I}$  and 0 otherwise. It follows that

$$P_A = U P_{U^\top A} U^\top = U I_{\mathcal{I}} U^\top = U_{\mathcal{I}} U_{\mathcal{I}}^\top$$

where  $U_{\mathcal{I}} = [u_{i_1}, \dots, u_{i_p}]$ . Thus  $P_A$  is the orthogonal projection onto the subspace spanned by the columns of  $U_{\mathcal{I}}$ . Since the column space of  $A$  coincides with the column space of  $U_{\mathcal{I}}$ , there exists an invertible  $p \times p$  matrix  $C$  such that  $A = U_{\mathcal{I}} C$ . Moreover,  $B = \hat{B}(A) = C^{-1} U_{\mathcal{I}} \Sigma_{YX} \Sigma_{XX}^{-1}$  and (8) and (9) are satisfied. There are  $\binom{n}{p}$  possible choices for  $\mathcal{I}$  and therefore  $\binom{n}{p}$  possible critical points with full rank. From (8), (9) and (10) results immediately.

To prove (11), use (c) to write:

$$\begin{aligned} E(A, B) &= (\text{vec}(Y - ABX))^\top (\text{vec}(Y - ABX)) \\ &= \text{vec}(Y)^\top \text{vec}(Y) - 2(\text{vec}(ABX))^\top \text{vec}(Y) + \text{vec}(ABX)^\top \text{vec}(ABX) \\ &= \text{tr}(YY^\top) - 2\text{tr}(ABXY^\top) + \text{tr}(ABXX^\top B^\top A^\top) \\ &= \text{tr}(\Sigma_{YY}) - 2\text{tr}(W\Sigma_{XY}) + \text{tr}(W\Sigma_{XX}W^\top) \end{aligned}$$

If  $A$  is full rank and  $B = \hat{B}(A)$ , then  $W = AB(A) = P_A \Sigma_{YX} \Sigma_{XX}^{-1}$  and therefore:

$$\begin{aligned} \text{tr}(W\Sigma_{XX}W^\top) &= \text{tr}(P_A \Sigma P_A) = \text{tr}(P_A \Sigma) = \text{tr}(U P_{U^\top A} U^\top U \Lambda U^\top) = \\ &= \text{tr}(P_{U^\top A} U^\top U \Lambda) = \text{tr}(P_{U^\top A} \Lambda) \end{aligned}$$

and

$$\text{tr}(W\Sigma_{YX}) = \text{tr}(P_A \Sigma) = \text{tr}(P_{U^\top A} \Lambda)$$



So, for an arbitrary  $A$  of rank  $p$ :

$$E(A, \hat{B}(A)) = \text{tr}(\Sigma_{YY}) - \text{tr}(P_{U^\top A} \Lambda)$$

If  $A$  is of the form  $U_{\mathcal{I}} C$ , then  $P_{U^\top A} = I_{\mathcal{I}}$ , therefore:

$$E(A, \hat{B}(A)) = \text{tr}(\Sigma_{YY}) - \text{tr}(I_{\mathcal{I}} \Lambda) = \text{tr}(\Sigma_{YY}) - \sum_{i \in \mathcal{I}} \lambda_i$$

which is (11).

We shall now establish that whenever  $A$  and  $B$  satisfy (8) and (9) with  $\mathcal{I} = 1, 2, \dots, p$  there exist matrices  $\bar{A}, \bar{B}$  arbitrarily close to  $A, B$  such that  $E(\bar{A}, \bar{B}) < E(A, B)$ . For this purpose it is enough to slightly perturb the column space of  $A$  in the direction of an eigenvector associated with one of the first  $p$  eigenvalues of  $\Sigma$  which is not contained in  $\{\lambda_i, i \in \mathcal{I}\}$ . More precisely, fix two indices  $j$  and  $k$  with  $j \in \mathcal{I}, k \notin \mathcal{I}$ . For any  $\epsilon$ , put:

$$\tilde{u}_j = (1 + \epsilon^2)^{-\frac{1}{2}}(u_j + \epsilon u_k) = \frac{1}{\sqrt{1 + \epsilon^2}}(u_j + \epsilon u_k)$$

and construct  $\tilde{U}_{\mathcal{I}}$  from  $U_{\mathcal{I}}$  by replacing  $u_i$  with  $\tilde{u}_j$ . Since  $k \notin \mathcal{I}$ , we still have  $\tilde{U}_{\mathcal{I}}^\top \tilde{U}_{\mathcal{I}} = I_p$ . Now let  $\tilde{A} = \tilde{U}_{\mathcal{I}} C$  and

$$\tilde{B} = \hat{B}(\tilde{A}) = C^{-1} \tilde{U}_{\mathcal{I}}^\top \Sigma_{YX} \Sigma_{XX}^{-1}$$

A simple calculation shows that the diagonal elements of  $P_{U^\top A}$  are:

$$\delta_i = \begin{cases} 0 & \text{if } i \notin \mathcal{I} \cup \{k\} \\ 1 & \text{if } i \in \mathcal{I} \text{ and } i \neq j \text{ and } i \neq k \\ \frac{1}{1 + \epsilon^2} & \text{if } i = j \\ \frac{\epsilon^2}{1 + \epsilon^2} & \text{if } i = k \end{cases}$$

Therefore:

$$\begin{aligned} E(\tilde{A}, \tilde{B}) &= \text{tr}(\Sigma_{YY}) - \text{tr}(P_{U^\top \tilde{A}} \Lambda) \\ &= \text{tr} \Sigma_{YY} - \left[ \sum_{i \in \mathcal{I} - \{j\}} \lambda_i + \frac{\lambda_j}{(1 + \epsilon^2)} + \frac{\epsilon^2 \lambda_k}{(1 + \epsilon^2)} \right] \\ &= \text{tr} \Sigma_{YY} - \sum_{i \in \mathcal{I}} \lambda_i - \frac{\epsilon^2 (\lambda_k - \lambda_j)}{(1 + \epsilon^2)} \\ &= E(A, B) - \frac{\epsilon^2 (\lambda_k - \lambda_j)}{(1 + \epsilon^2)}. \end{aligned}$$

By taking values of  $\epsilon$  arbitrarily small, we see that any neighborhood of  $A, B$  contains points of the form  $\tilde{A}, \tilde{B}$  with a strictly smaller error function. Thus if  $\mathcal{I} \neq \{1, 2, \dots, p\}$ , then (8) and (9) define a saddle point and not a local minimum. Notice that, in any case, it could not be a local maximum because of the strict convexity of  $E$ , with fixed full rank  $A$ , in fact 1.

close to 0, we can make  $E(\tilde{A}, \tilde{B})$  arbitrarily close to  $E(A, B)$  and strictly smaller. This shows that the critical point defined by  $A$  and  $B$  is a saddle point. The proof of fact 4 is now complete.

## 2.6 Equivalence with PCA

In this section we show in detail the equivalence between the principal component analysis (PCA) and the linear autoencoder. We consider the case of centered data, that is, the mean of each column of the data matrix  $X$  is zero. The covariance matrix of the data is then given by  $\Sigma_{XX} = \frac{1}{N}XX^\top$ . The principal component analysis consists in finding the orthogonal matrix  $U$  such that the first  $p$  columns of  $U$  are the eigenvectors of  $\Sigma_{XX}$  associated with the  $p$  largest eigenvalues. The matrix  $U$  is then used to project the data onto the subspace spanned by the first  $p$  principal components. The linear autoencoder consists in finding the matrices  $A$  and  $B$  such that  $W = AB$  is the orthogonal projection onto the subspace spanned by the first  $p$  eigenvectors of  $\Sigma_{XX}$  associated with the  $p$  largest eigenvalues. The matrix  $A$  is then used to project the data onto the subspace spanned by the first  $p$  principal components.

We are now going to show that if we fix a linear decoder and a squared error loss, the optimal solution is obtained when using a linear encoder. We denote:

- $X = [x_1, \dots, x_m]$  the data matrix
- $W^*$  the fixed linear decoder matrix
- $W$  the linear encoder matrix
- $H = XW$  the hidden layer

The aim is to minimize the reconstruction error, i.e. the difference between the input data and the output data which can be written as:

$$\min_{\theta} \sum_{i=1}^m \sum_{j=1}^n (x_{ij} - \hat{x}_{ij})^2 \equiv \min_{HW^*} (\|X - HW^*\|_F)^2$$

But the problem of finding a matrix with minimum distance to another matrix is trivial since the **Eckart-Young theorem** guarantees that:

**Theorem 2.1.** *For either  $\|\cdot\|_F$ ,  $\|\cdot\|_2$  we have:*

$$\|A - A_k\| \leq \|A - B\| \quad \forall B \text{ of rank } k$$

So, if the hidden layer of the autoencoder is  $k$ , the optimal solution is the truncated SVD:

$$HW^* = U_{:, \leq k} \Sigma_{k,k} V_{:, \leq k}^\top$$

By matching variables one possible solution is:

$$H = U_{:, \leq k} \Sigma_{k,k} \quad W^* = V_{:, \leq k}^\top$$

We are now going to show that  $H$  is a linear encoding and find an expression for the encoder weights  $W$ :

*Proof.*

$$\begin{aligned}
H &= U_{:, \leq k} \Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1} U_{:, \leq K} \Sigma_{k,k} && \text{(pre-multiplying } (XX^T)(XX^T)^{-1} = I) \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1} U_{:, \leq k} \Sigma_{k,k} && \text{(using } X = U\Sigma V^T) \\
&= XV\Sigma^T U^T (U\Sigma\Sigma^T U^T)^{-1} U_{:, \leq k} \Sigma_{k,k} && (V^T V = I) \\
&= XV\Sigma^T U^T U (\Sigma\Sigma^T)^{-1} U^T U_{:, \leq k} \Sigma_{k,k} && ((ABC)^{-1} = C^{-1}B^{-1}A^{-1}) \\
&= XV\Sigma^T (\Sigma\Sigma^T)^{-1} U^T U_{:, \leq k} \Sigma_{k,k} && (U^T U = I) \\
&= XV\Sigma^T \Sigma^{T^{-1}} \Sigma^{-1} U^T U_{:, \leq k} \Sigma_{k,k} && ((AB)^{-1} = B^{-1}A^{-1}) \\
&= XV\Sigma^{-1} I_{:, \leq k} \Sigma_{k,k} && (U^T U_{:, \leq k} = I_{:, \leq k}) \\
&= X V I_{:, \leq k} && (\Sigma^{-1} I_{:, \leq k} = \Sigma_{k,k}^{-1}) \\
H &= X V_{:, \leq k}
\end{aligned}$$

Thus  $H$  is a linear transformation of  $X$  and  $W = V_{:, \leq k}$ . The encoder matrix is the matrix of the first  $k$  eigenvectors of the data covariance matrix. This means that the hidden layer of the autoencoder is just the principal component analysis of the data.  $\square$

### 3 Boolean Autoencoders

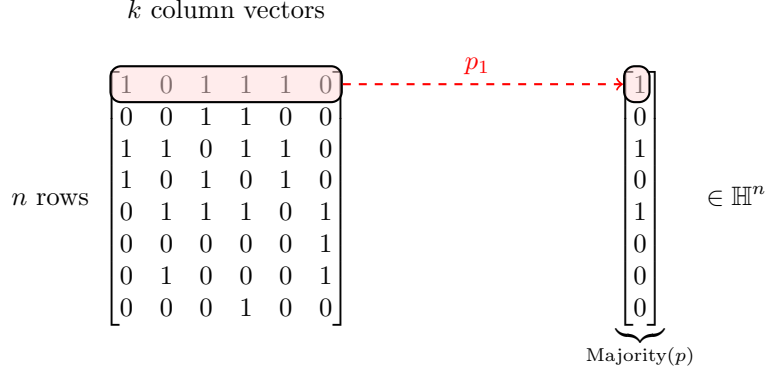
Boolean autoencoders correspond to the case where  $\mathbb{F} = \mathbb{G} = \{0, 1\}$ ,  $A$  and  $B$  are classes of Boolean functions, and  $\Delta$  is the Hamming distance. Traditionally, a Boolean function is defined as a mapping from  $\{0, 1\}^n$  to  $\{0, 1\}$ . but here we use the same term more generally to refer to Boolean vector functions, that is functions from  $\{0, 1\}^n$  to  $\{0, 1\}^m$  which of course can be seen as  $m$  Boolean functions.

$$\mathbb{H}^p \ni \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 1 \end{bmatrix} \in \mathbb{H}^m$$

In the general framework, the sets  $\mathcal{A}, \mathcal{B}$  contain all possible Boolean functions of the right dimensions. Given  $k$  binary column vectors  $p_1, \dots, p_k$  in the  $n$ -dimensional hypercube  $\mathbb{H}^n$ , we define the corresponding binary majority vector  $\text{Majority}(p)$  in  $\mathbb{H}^n$  by taking in each row  $j$  the majority of the corresponding components  $p_{ij}$ . When  $k^1$  is even, there can be ties in which case one can flip a fair coin to assign the corresponding value.

---

<sup>1</sup>The original paper states that one can flip a fair coin to assign the majority in ties if  $n$  is even. I think it is an error and the tie can occur if  $k$  is even since that is the number of elements considered, not  $n$ .



**Lemma 3.1.** *The vector  $\text{Majority}(p)$  is a vector in  $\mathbb{H}^n$  closest to the center of gravity of the vectors  $p_1, \dots, p_k$  and it minimizes the function  $E(q) = \sum_{i=1}^k \Delta(p_i, q)$ .*

*Proof.* The center of gravity is the vector  $c$  in  $\mathbb{R}^n$  with coordinates

$$c_j = \frac{\left( \sum_{i=1}^k p_{ji} \right)}{k}$$

So, each  $c_j$  is the average of the  $j$ -th components of the vectors  $p_1, \dots, p_k$ . For any  $j$ ,  $(p)_j$  is the closest binary value to  $c_j$ . Furthermore:

$$\sum_{i=1}^k \Delta(\text{Majority}(p), p_i) = \sum_{i=1}^k \sum_{j=1}^n \Delta(\text{Majority}(p)_j, p_{ij}) = \sum_{j=1}^n \left( \sum_{i=1}^k \Delta(\text{Majority}(p)_j, p_{ij}) \right)$$

and each term in the last sum is minimized by the majority vector. □

A **Voronoi partition** of  $\mathbb{H}^n$  generated by the vectors  $p_1, \dots, p_k$  is a partition of  $\mathbb{H}^n$  into  $k$  regions  $\mathcal{C}^{Vor}(p_1), \dots, \mathcal{C}^{Vor}(p_k)$  such that for each  $x$  in  $\mathbb{H}^n$ :

$$x \in \mathcal{C}^{Vor}(p_i) \iff \Delta(x, p_i) \leq \Delta(x, p_j) \text{ for all } j \neq i$$

And this can be visualized as:

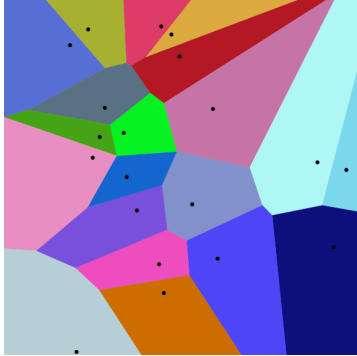


Figure 3: Voronoi diagram with Euclidean distance metric

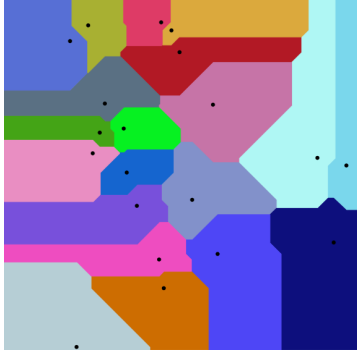


Figure 4: Voronoi diagram with Manhattan distance metric

Recall that autoencoders performs a mapping of the input space  $\mathbb{H}^n$  into a lower-dimensional space  $\mathbb{H}^p$  and then back to the original space. The mapping is done by two functions  $A$  and  $B$  that are learned from the data. The mapping is done in two steps:

$$x \xrightarrow{B} h \xrightarrow{A} y$$

**Theorem 3.2. Fixed layer solution:** *if the  $A$  mapping is fixed, then the optimal mapping  $B^*$  is given by  $B^*(x) = h_i$  for any  $x$  in  $\mathcal{C}_i = \mathcal{C}^{Vor}(A(h_i))$ . Conversely, if  $B$  is fixed, then the optimal mapping  $A^*$  is given by  $A^*(h_i) = \text{Majority}[\mathcal{X} \cap B^{-1}(h_i)]$*

*Proof.* Assume first that  $A$  is fixed. Then for each of the  $2^p$  possible Boolean vectors  $h_1, \dots, h_{2^p}$  of the hidden layer,  $A(h_1), \dots, A(h_{2^p})$  provide  $2^p$  points (centroids) in the hypercube  $\mathbb{H}^n$ . One can build the corresponding Voronoi partition by assigning each point of  $\mathbb{H}^n$  to its closest centroid, breaking ties arbitrarily, thus forming a partition of  $\mathbb{H}^n$  into  $2^p$  corresponding clusters  $\mathcal{C}_1, \dots, \mathcal{C}_{2^p}$ , with  $\mathcal{C}_i = \mathcal{C}^{Vor}(A(h_i))$ . The optimal mapping  $B^*$  is then given by  $B^*(x) = h_i$  for any  $x$  in  $\mathcal{C}_i$ .

If we consider the input-output layers to have a cardinality of 4 and the hidden layer to have a cardinality of 2, then this would mean that there would be  $2^2 = 4$  centroids given by the  $A$  mapping in the space  $\mathbb{H}^4$  showed in figure:

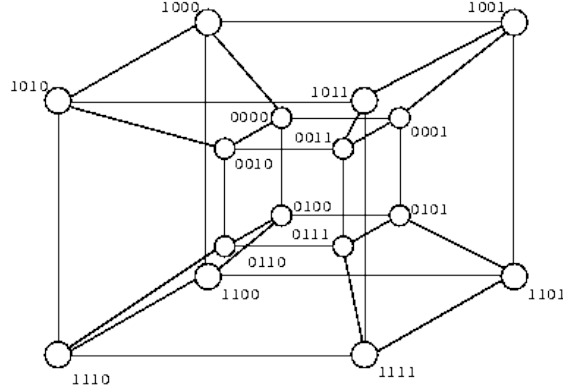


Figure 5: Hypercube in 4 dimensions

So, the Voronoi partition would be the partition of the hypercube into 4 regions using as metric the Hamming distance, i.e. the number of edges that need to be crossed to go from one point to each centroid.

Conversely, assume that  $B$  is fixed. Then for each of the  $2^p$  possible Boolean vectors  $h_1, \dots, h_{2^p}$  of the hidden layer, let  $B^{-1}(h_i) = \{x \in \mathbb{H}^n : B(x) = h_i\}$ . To minimize the reconstruction error,  $A^*$  must map  $h_i$  onto a point  $y$  of  $\mathbb{H}^n$  minimizing the sum of Hamming distances to points in  $\mathcal{X} \cap B^{-1}(h_i)$ . By Lemma 3.1 the minimum is realized by the component-wise majority vector  $A^*(h_i) = \text{Majority}[\mathcal{X} \cap B^{-1}(h_i)]$ , breaking ties arbitrarily. Note that this solution minimizes the distortion on the training set. The generalization or total distortion however, is minimized by  $A^*(h_i) = \text{Majority}[B^{-1}(h_i)]$ . In some situations, one may have the additional constraint that the output vector must belong to the training. With this additional constraint the optimal solution is  $A^*(h_i)$  should be the vector  $\mathcal{X}$  that is closest to the vector  $\text{Majority}[\mathcal{X} \cap B^{-1}(h_i)]$ .  $\square$

## 4 Clustering complexity on the hypercube

### 4.1 Brief recap on computational complexity

Recall that  $\mathcal{P}$  is the class of problems that can be *solved* in polynomial time by a deterministic Turing machine, and  $\mathcal{NP}$  is the class of problems for which a solution can be *verified* in polynomial time by a deterministic Turing machine. The class  $\mathcal{NP}$  is the class of problems that can be solved non-deterministically in polynomial time. A problem is:

- **$\mathcal{NP}$ -complete** if it is in  $\mathcal{NP}$  and every problem in  $\mathcal{NP}$  can be reduced to it in polynomial time
- **$\mathcal{NP}$ -hard** if there is a  $\mathcal{NP}$ -complete problem that can be reduced to it in polynomial time

An example of an  $\mathcal{NP}$ -complete problem is the Boolean satisfiability problem (SAT), which is the problem of determining whether a given Boolean formula can be satisfied by assigning truth values to its variables. If it could be solved in polynomial time, then every problem in  $\mathcal{NP}$  could be solved in polynomial time because every problem in  $\mathcal{NP}$  can be reduced to SAT in polynomial time.

## 4.2 Clustering complexity

In this section, we briefly review some results on clustering complexity and then prove that the hypercube clustering decision problem is in general NP-complete.

A graph is cubical if it is the subgraph of some hypercube  $\mathbb{H}^d$  for some  $d$ . Although deciding whether a graph is cubical is NP-complete, there is a theorem providing a necessary and sufficient condition for a graph to be cubical. A graph  $G(V, E)$  is cubical and embeddable in  $\mathbb{H}^d$  if and only if it is possible to color the edges of  $G$  with  $d$  colors such that:

1. All edges incident with a common vertex are of different color
2. In each path of  $G$ , there is some color that appears an odd number of times
3. In each cycle of  $G$ , no color appears an odd number of times

**Theorem 4.1.** *Consider the following hypercube clustering problem:*

- **Input:**  $m$  binary vectors  $x_1, \dots, x_m$  of length  $n$  and an integer  $k$
- **Output:**  $k$  binary vectors  $c_1, \dots, c_k$  of length  $n$  (the centroids) and a function  $f$  from  $\{x_1, \dots, x_m\}$  to  $\{c_1, \dots, c_k\}$  that minimizes the distortion

$$E = \sum_{t=1}^m \Delta(x_t, f(x_t))$$

Where  $\Delta$  is Hamming distance.

The hypercube clustering decision problem  $\mathcal{NP}$ -hard when  $k \sim m^\epsilon$  ( $\epsilon > 0$ )

*Proof.* To sketch the reduction, we start from the problem of clustering  $m$  points in the plane  $\mathbb{R}^2$  using cluster centroids and the  $L_1$  distance, which is  $\mathcal{NP}$ -complete by reduction from 3-SAT when  $k \sim m^\epsilon$  ( $\epsilon > 0$ ). Without any loss of generality, we can assume that the points in these problems lie on the vertices of a square lattice. Using the theorem in Havel and Moravek, one can show that a  $n \times m$  square lattice in the plane can be embedded in a hypercube  $\mathbb{H}^{m+n}$ . Example:



It is easy to check that the  $L_1$  or Manhattan distance between any two points on the square lattice is equal to the corresponding Hamming distance in  $\mathbb{H}^{m+n}$ . This polynomial reduction completes the proof that if the number of cluster satisfies  $k = 2^p \sim m^\epsilon$  or equivalently  $p \sim \epsilon \log_2 m \sim C \log n$ , then the hypercube clustering problem associated with the Boolean autoencoder is  $\mathcal{NP}$ -hard and the corresponding decision problem is  $\mathcal{NP}$ -complete. If the numbers  $k$  of clusters is fixed and the centroids

must belong to the training set, there are only  $\binom{m}{k} \sim m^k$  possible choices for the centroids inducing the corresponding Voronoi clusters. This yields a trivial, albeit not efficient, polynomial time algorithm. When the centroids are not required to be in the training set, we conjecture also the existence of polynomial time algorithms by adapting the corresponding theorems in Euclidean space.  $\square$

## 5 Overcomplete Autoencoders

An autoencoder is said to be overcomplete if the number of neurons in the hidden layer is greater than the number of neurons in the input layer and  $\mathbb{F} = \mathbb{G}$ . This is a common practice in the field of deep learning, where the hidden layer is used to learn a representation of the input data. The idea behind overcomplete autoencoders is that by using more neurons in the hidden layer, the network can learn a more complex representation of the input data, which can lead to better performance on certain tasks.

Of course there is an optimal solution, the identity function thus this case is interesting only if additional constraints such as regularization and restrictions on  $\mathcal{A}, \mathcal{B}$  are added to the problem.

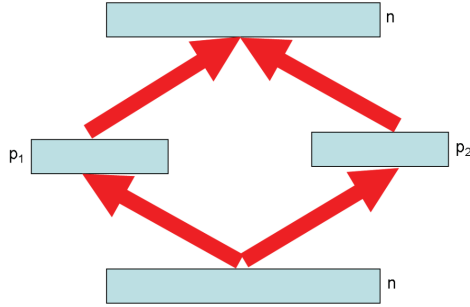


Figure 6: Horizontally combining autoencoder

In this context of large hidden layers, in addition to vertical composition, there is also a natural horizontal composition for autoencoders that can be used to create large hidden layer representations simply by horizontally combining autoencoders. Two (or more) autoencoders with architectures  $n/p_1/n$  and  $n/p_2/n$  can be trained and the hidden layers can be combined to yield an expanded hidden representation of size  $p_1 + p_2$  that can then be fed to the subsequent layers of the overall architecture. Differences in the  $p_1$  and  $p_2$  hidden representations could be introduced by many different mechanisms, for instance using different learning algorithms, different initializations, different training samples, different learning rates, or different distortion measures.