Autoencoders

Lorenzo Bozzoni

Politecnico di Milano

September 2024

Table of content

1. General framework

2. Linear autoencoders

3. Boolean autoencoders

4. Clustering complexity

General framework

The general framework of autoencoders is:

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

Where $B \in \mathcal{B}$ which is a set of functions from $\mathbb{F}^n$ to $\mathbb{G}^p$ while $A \in \mathcal{A}$ which is a set of functions from $\mathbb{G}^p$ to $\mathbb{F}^n$.

General framework

The goal is to find a pair of functions A, B such that the generic dissimilarity function Δ is minimized:

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

In the auto-associative case the right side of the autoencoder is again x_t.
The focus is not on the reconstruction of the input but rather on how well we can compress the input data in the hidden layer without losing information.

Linear autoencoders

In the case of **linear autoencoders** we have:

- $\mathbb{F}, \mathbb{G}$ are fields
- $\mathcal{A}, \mathcal{B}$ are the classes of linear transformations: A, B are respectively matrices of shape $p \times n$ and $n \times p$
- Δ is the squared Euclidean distance (L_2^2 norm)

Linear autoencoders

In general the problem of finding the matrices A, B that minimize the error function E is a non-convex optimization problem.

However, fixing one of the two matrices, the problem becomes convex so **we can find the optimal value by alternating the optimization of the two matrices.** Fixing A the optimal B is:

$$B = \hat{B}(A) = (A^\intercal A)^{-1} A^\intercal$$

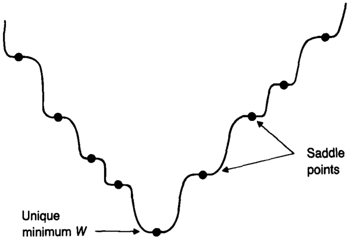
While fixing B the optimal A is:

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

Where Σ_{XX} is the covariance matrix of the input data.

Linear autoencoders

An important result is the shape of the error function E:



- Σ is full-rank with n distinct eigenvalues $\lambda_1 > \cdots > \lambda_n$
- $\mathcal{I} = i_1, \ldots, i_p$ $(1 \leq i_1 < \cdots < i_p \leq n)$ is any ordered p-index set
- $U_{\mathcal{I}} = [u_{i_1}, \ldots, u_{i_p}]$ matrix formed by the orthonormal eigenvectors of Σ associated with the eigenvalues $\lambda_{i_1}, \ldots, \lambda_{i_p}$

The critical map W associated with the index set $\{1, 2, \ldots, p\}$ 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 Σ with $q < p$

Linear autoencoders

Since we are applying only linear transformations, the best compression we can achieve is the one that projects the input data on the subspace spanned by the eigenvectors of the covariance matrix of the input data.

This corresponds to the **Principal Component Analysis (PCA)** when the input is normalized as follows:

$$\hat{x}_{i,j} = \frac{1}{\sqrt{m}} \left(x_{ij} - \frac{1}{m} \sum_{k=1}^{m} x_{kj} \right)$$

Linear autoencoders

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

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

From the Eckart-Young theorem, we know that the optimal solution is the truncated SVD:

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

By matching variables one possible solution is:

$$H = U_{:,\leq k} \Sigma_{k,k} \qquad W^* = V_{:,\leq k}^{\mathsf{T}}$$

Linear autoencoders

Proof.

$$
\begin{aligned}
H &= U_{:,\leq k}\Sigma_{k,k} \\
&= (XX^T)(XX^T)^{-1}U_{:,\leq K}\Sigma_{k,k} \\
&= (XV\Sigma^T U^T)(U\Sigma V^T V\Sigma^T U^T)^{-1}U_{:,\leq k}\Sigma_{k,k} \\
&= XV\Sigma^T U^T(U\Sigma\Sigma^T U^T)^{-1}U_{:,\leq k}\Sigma_{k,k} \\
&= XV\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T U_{:,\leq k}\Sigma_{k,k} \\
&= XV\Sigma^T(\Sigma\Sigma^T)^{-1}U^T U_{:,\leq k}\Sigma_{k,k} \\
&= XV\Sigma^T\Sigma^{T^{-1}}\Sigma^{-1}U^T U_{:,\leq k}\Sigma_{k,k} \\
&= XV\Sigma^{-1}I_{:,\leq k}\Sigma_{k,k} \\
&= XVI_{:,\leq k} = XV_{:,\leq k}
\end{aligned}
$$

Thus H is a linear transformation of X and the encoder matrix is the matrix of the first k eigenvectors of the data covariance matrix. □

Boolean autoencoders

In the case of **Boolean autoencoders** we have:

- $\mathbb{F}, \mathbb{G}$ are the Boolean fields, i.e $\{0, 1\}$, the Galois field $\mathbb{F}_2$
- $\mathcal{A}, \mathcal{B}$ are the classes of Boolean transformations: A, B are unrestricted boolean functions
- Δ is the Hamming distance

Boolean autoencoders

We start defining the following lemma:

Lemma

The vector 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 $E(q) = \sum_{i=1}^{k} \Delta(p_i, q)$.

Where 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.

Boolean autoencoders

This means that for each row, we check the majority of the values and we set the value of the row to the majority value.

$$n \text{ rows} \quad \overset{k \text{ column vectors}}{\begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}} \xrightarrow{p_1} \underbrace{\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{\text{Majority}(p)} \in \mathbb{H}^n$$

Boolean autoencoders

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

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

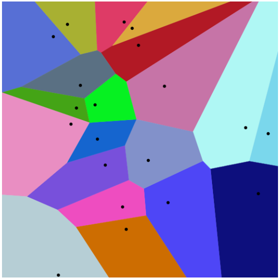


Figure: Euclidean distance

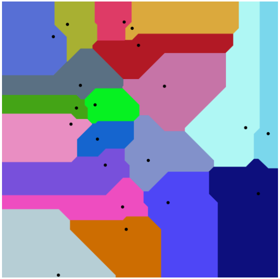


Figure: Manhattan distance

Boolean autoencoders

Considering the two steps mapping:

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

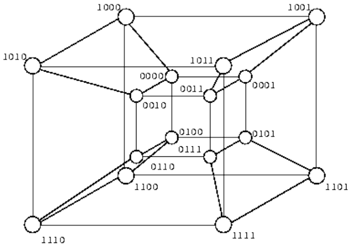
Where the dimension of the hidden layer is p, so there are 2^p possible configurations of the hidden layer, denoted by $h_1, \dots, h_{2^p}$.

Theorem

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 $A^*(h_i) = Majority\left[\mathcal{X} \cap B^{-1}(h_i)\right]$*

Boolean autoencoders

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:



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.

Complexity recap

The basic classes for complexity are:

- $\mathcal{P}$ is the class of problems that can be *solved* in polynomial time by a deterministic TM
- $\mathcal{NP}$ is the class of problems for which a solution can be *verified* in polynomial time by a deterministic TM. The class $\mathcal{NP}$ is the class of problems that can be solved non-deterministically in polynomial time
  - $\mathcal{NP}$-**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

Clustering complexity

Theorem

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*

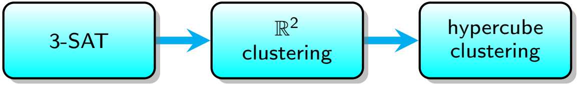
$$E = \sum_{t=1}^{m} \Delta(x_t, f(x_t))$$

Where Δ is Hamming distance.

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

Clustering complexity

To prove the hypercube clustering problem is $\mathcal{NP}$-hard we need to demonstrate that an $\mathcal{NP}$-complete problem can be reduced to it in polynomial time. The following reductions are used:


3-SAT

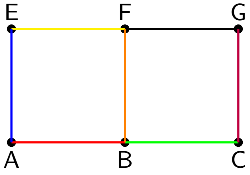
$\mathbb{R}^2$ clustering

hypercube clustering

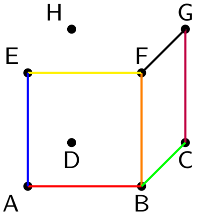
To sketch the reduction, we start from the problem of clustering m points in the plane $\mathbb{R}^2$ using cluster centroids and the L_1 distance, which is $\mathcal{NP}$-complete by reduction from 3-SAT when $k \sim m^\epsilon$ $(\epsilon > 0)$. Without any loss of generality, we can assume that the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

Clustering complexity

Example:



2x1 Lattice



(2+1)D Cube

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.

Thank you for your attention!