Des. Codes Cryptogr. (2012) 65:383–403
DOI 10.1007/s10623-012-9719-x

Boolean autoencoders and hypercube clustering complexity

P. Baldi

Received: 1 November 2010 / Revised: 5 June 2012 / Accepted: 18 June 2012 /
Published online: 24 July 2012

Abstract We introduce and study the properties of Boolean autoencoder circuits. In particular, we show that the Boolean autoencoder circuit problem is equivalent to a clustering problem on the hypercube. We show that clustering m binary vectors on the n-dimensional hypercube into k clusters is NP-hard, as soon as the number of clusters scales like m^{ϵ} ($\epsilon > 0$), and thus the general Boolean autoencoder problem is also NP-hard. We prove that the linear Boolean autoencoder circuit problem is also NP-hard, and so are several related problems such as: subspace identification over finite fields, linear regression over finite fields, even/odd set intersections, and parity circuits. The emerging picture is that autoencoder optimization is NP-hard in the general case, with a few notable exceptions including the linear cases over infinite fields or the Boolean case with fixed size hidden layer. However learning can be tackled by approximate algorithms, including alternate optimization, suggesting a new class of learning algorithms for deep networks, including deep networks of threshold gates or artificial neurons.

Keywords Autoencoders · Clustering · Boolean circuits · Computational complexity

Mathematics Subject Classification (2010) 68T05

1 Introduction

Autoencoder circuits, which try to minimize a distortion measure between inputs and outputs, play a fundamental role in machine learning. An autoencoder is a feedforward circuit with n input gates, p intermediary or hidden gates, and n output gates. In broad terms, given a set $\mathcal{X}$ of training input vectors, the goal is to select appropriate input-to-hidden and

This is one of several papers published together in *Designs, Codes and Cryptography* on the special topic: “Combinatorics – A Special Issue Dedicated to the 65th Birthday of Richard Wilson”.

P. Baldi (✉)
Department of Computer Science, University of California, Irvine, CA, USA
e-mail: pfbaldi@ics.uci.edu

hidden-to-output transformation functions so as to minimize a distortion measure between inputs vectors and the corresponding output vectors produced by the circuit. Autoencoders were introduced in the 1980s by the Parallel Distributed Processing group [23] as a way to address the problem of unsupervised learning, learning from data in the absence of any target, by using the data itself as the output target. When $p < n$, the hidden layer acts as a bottleneck forcing the circuit to extract features and produce a compressed representations of the input data $\mathcal{X}$. More recently, autoencoders have been used extensively in the "deep architecture" approach [16,17,5,10], where autoencoders in the form of Restricted Boltzman Machines (RBMS) are stacked and trained bottom up in unsupervised fashion to extract hidden features and efficient representations that can then be used to address supervised classification or regression tasks.

In spite of the interest they have generated, and with few exceptions [3,26], little theoretical understanding of autoencoders and deep architectures has been obtained to this date, especially in the non-linear case. Here we derive a better theoretical understanding of autoencoders in part by studying an extreme form of non-linear autoencoder, namely the Boolean autoencoder.

2 The autoencoder problem

2.1 Autoencoder problem

We begin with a fairly general definition of the autoencoder problem. An $n/p/n$ autoencoder (Fig. 1) is defined by a t-uple $n, p, m, \mathbb{F}, \mathbb{G}, \mathcal{A}, \mathcal{B}, \mathcal{X}, \Delta$ where:

1. n, p and m are positive integers. Here we consider primarily the case where $0 < p < n$.
2. $\mathbb{F}$ and $\mathbb{G}$ are sets.
3. $\mathcal{A}$ is a class of functions from $\mathbb{G}^p$ to $\mathbb{F}^n$.
4. $\mathcal{B}$ is a class of functions from $\mathbb{F}^n$ to $\mathbb{G}^p$.
5. $\mathcal{X} = \{x_1, \dots, x_m\}$ is a set of m training vectors in $\mathbb{F}^n$. When external targets are present, we let $\mathcal{Y} = \{y_1, \dots, y_m\}$ denote the corresponding set of target vectors in $\mathbb{F}^n$.
6. Δ is a distance or distortion function (e.g. L_p norm, Hamming distance) 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 $A \circ B(x) \in \mathbb{F}^n$ (Fig. 1). The corresponding *autoencoder problem* is to find $A \in \mathcal{A}$ and $B \in \mathcal{B}$ that minimize the overall distortion or error function:

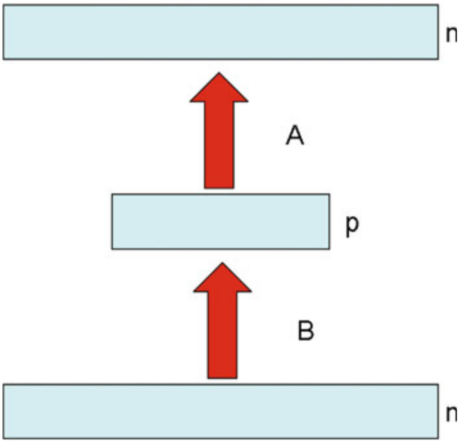

Fig. 1 An n/p/n autoencoder circuit (Color figure online)

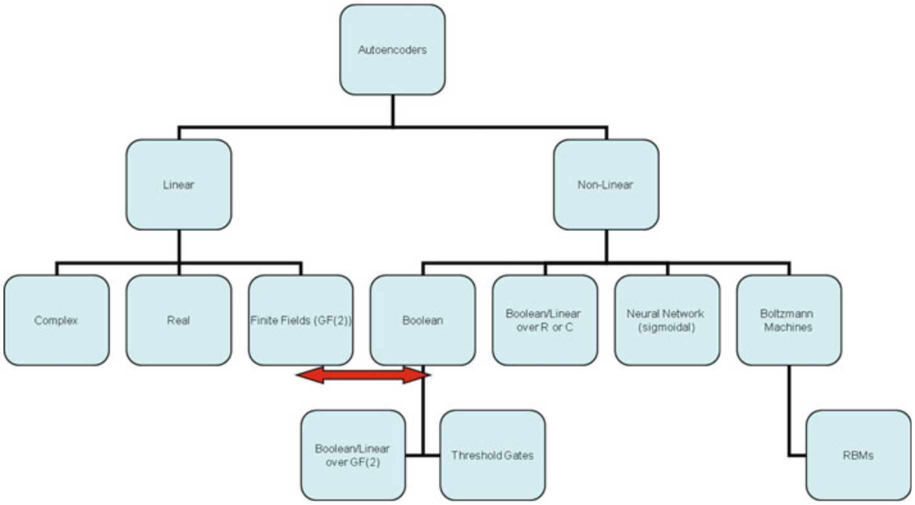

Fig. 2 Simple classification of autoencoder types (Color figure online)

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

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

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

Here we consider primarily the auto-associative case with $p < n$ where the goal of the autoencoder is to find a way to compress the data. The regime $p \geq n$ is also of interest, but requires in general additional assumptions beyond the scope of this article (see Discussion).

Obviously, from this general framework, different kinds of autoencoders can be derived depending, for instance, on the choice of sets $\mathbb{F}$ and $\mathbb{G}$, transformation classes $\mathcal{A}$ and $\mathcal{B}$, distortion function Δ, as well as the presence of any additional constraints (Fig. 2). Linear autoencoders correspond to the case where $\mathbb{F}$ and $\mathbb{G}$ are fields and $\mathcal{A}$ and $\mathcal{B}$ are the classes of linear transformations, hence A and B are matrices of size $p \times n$ and $n \times p$ respectively. In this case, the autoencoder problem can be viewed essentially as the problem of finding a rank p approximation to the identity function. The linear real-valued case where $\mathbb{F} = \mathbb{G} = \mathbb{R}$ and Δ is the squared Euclidean distance was addressed in [3]. Similar results were obtained more recently for the linear complex-valued case [4]. The main goal of this article is to study Boolean autoencoders where $\mathbb{F} = \mathbb{G} = \{0, 1\}$, including linear autoencoders over GF(2) $= \mathbb{F}_2$.

2.2 Autoencoder properties

From the study of different kinds of autoencoders [4,2] emerge a set of basic properties that ought to be investigated for each class of autoencoders.

(1) Invariances. What are the relevant group actions for the problem? What are the transformations of $\mathbb{F}^n$ and $\mathbb{G}^p$, or A and B, that leave the problem invariant?
(2) Fixed layer solutions. Is it possible to optimize A (resp. B), fully or partially, while B (resp. A) is held constant?
(3) Problem complexity. How complex is the autoencoder optimization problem? Is there an overall analytical solution? Is the corresponding decision problem NP-complete?
(4) Landscape of E. What is the landscape of the overall error E? Are there any symmetries, local minima, critical points and how can they be characterized?
(5) Clustering. Especially in the case where $p < n$, what is the relationship to clustering?
(6) Transposition. Is there a notion of symmetry or transposition between the transformations A and B, in particular around critical points?
(7) Recycling. What happens if the values from the output layer are recycled into the input layer, in particular around critical points?
(8) Learning algorithms. What are the learning algorithms and their properties? In particular, can A and B be fully, or partially, optimized in alternation? And if so, is the algorithm convergent? And if so, at what speed and what are the properties of the corresponding limit points?
(9) Generalization. What are the generalization properties of the autoencoder after learning? In other words, what are the properties of the distortion function (e.g. its average) on vectors in $\mathbb{F}^n - \mathcal{X}$?
(10) External targets. How does the problem change if external targets are provided?
(11) Composition. Autoencoder circuits can be stacked vertically, using the hidden layer of the autoencoder at one level of the stack as the input layer for the autoencoder at the next level of the stack (Fig. 4). What is the overall effect of such composition? Autoencoders can also be composed horizontally [2]. What is the overall effect of such composition?

Most of these questions can be addressed analytically in the case of real-valued or complex-valued autoencoders with the squared Euclidean distance ($\Delta = L_2^2$) as the distortion function. For completeness, we restate without proof the main results derived in [3] for the linear real-valued case and generalized in [4] for the complex-valued case.

3 Linear autoencoders over the real or complex numbers

We use A^t to denote the transpose of any matrix A in the real-valued case, or its conjugate transpose in the complex-valued case.
(1) Invariances.
(a) Change of coordinates in the hidden layer. Note that for any invertible $p \times p$ matrix C, we have $W = AB = ACC^{-1}B$ and $E(A, B) = E(AC, C^{-1}B)$. Thus all the properties of the linear autoencoder are fundamentally invariant with respect to any change of coordinates in the hidden layer.
(b) Change of coordinates in the input/output layers. Consider an orthonormal change of coordinates in the output space defined by an orthogonal (or unitary) $n \times n$ matrix D, and any change of coordinates in the input space defined by an invertible $n \times n$ matrix C. This leads to a new autoencoder problem with input vectors $Cx_1, \ldots, Cx_m$ and target output vectors of the form $Dy_1, \ldots, Dy_m$ with reconstruction error of the form

$$E(A', B') = \sum_t ||Dy_t - A'B'Cx_t||^2 \qquad (3)$$

If we use the one-to-one mapping between pairs of matrices (A, B) and (A', B') defined by $A' = DA$ and $B' = BC^{-1}$, we have

$$E(A', B') = \sum_t ||Dy_t - A'B'Cx_t||^2 = \sum_t ||Dy_t - DABx_t||^2$$

$$= \sum_t ||y_t - ABx_t||^2 \tag{4}$$

the last equality using the fact that D is an isometry and preserves distances and angles. Thus, using the transformation $A' = DA$ and $B' = BC^{-1}$ the original problem and the transformed problem are equivalent and the functions $E(A, B)$ and $E(A', B')$ have the same landscape. In particular, in the auto-associative case, we can take $C = D$ to be a unitary matrix. This leads to an equivalent autoencoder problem with input vectors Cx_t and covariance matrix $C\Sigma_{XX}C^{-1}$, with $\Sigma_{XX} = \sum_i x_i x_i^t$. For the proper choice of C, there is an equivalent problem where the basis of the space is provided by the eigenvectors of the covariance matrix and the covariance matrix is a diagonal matrix with diagonal entries equal to the eigenvalues of the original covariance matrix Σ.

(2) Fixed layer solutions. The problem becomes convex if A is fixed, or if B is fixed. When A is fixed, assuming A has rank p and that the data covariance matrix $\Sigma_{XX} = \sum_i x_i x_i^t$ is invertible, then at the optimum $B^* = B(A) = (A^t A)^{-1} A^t$. When B is fixed, assuming B has rank p and that Σ_{XX} is invertible, then at the optimum $A^* = A(B) = \Sigma_{XX} B^t (B \Sigma_{XX} B^t)^{-1}$.

(3) Problem complexity. While the cost function is quadratic and all the operations are linear, the overall problem is not convex because the hidden layer limits the rank of the overall transformation to be at most p, and the set of matrices of rank p or less is *not* convex. However the linear autoencoder problem over $\mathbb{R}$ and $\mathbb{C}$ can be solved analytically.

(4) Landscape of E. The overall landscape of E has no local minima. All the critical points where the gradient of E is zero, correspond to projections onto subspaces associated with subsets of eigenvectors of the covariance matrix Σ_{XX}. Projections onto the subspace associated with the p largest eigenvalues correspond to the global minimum and Principal Component Analysis. All other critical points, corresponding to projections onto subspaces associated with other set of eigenvalues, are saddle points (Fig. 3). More precisely, if $\mathcal{I} = i_1, \ldots, i_p$ $(1 \leq i_i < \ldots < i_p \leq n)$ is any ordered list of indices, let $U_{\mathcal{I}} = [u_1, \ldots, u_p]$ denote the matrix formed by the orthonormal eigenvectors of Σ_{XX} associated with the eigenvalues $\lambda_{i_1}, \ldots, \lambda_{i_p}$. Then two matrices A and B of rank p define a critical point if and only if there is a set $\mathcal{I}$ and an invertible $p \times p$ matrix C such that $A = U_{\mathcal{I}}C$, $B = C^{-1}U_{\mathcal{I}}^t$, and $W = AB = P_{U_{\mathcal{I}}}$, where $P_{U_{\mathcal{I}}}$ is the orthogonal projection onto the subspace spanned by the columns of $U_{\mathcal{I}}$. At the global minimum, assuming that $C = I$, the activities in the hidden layer are given by the dot products $u_1^t x \ldots u_p^t x$ and correspond to the coordinates of x along the first p eigenvectors of Σ_{XX}.

(5) Clustering. The global minimum performs a form of clustering by hyperplane, with respect to $KerB$, the kernel of B. For any given vector x, all the vectors of the form $x + Ker(B)$ are mapped onto the same vector $y = AB(x) = AB(x + KerB)$.

(6) Transposition. Symmetries and Hebbian Rules. At the global minimum, for $C = I$, $A = B^t$.

(7) Recycling. At any critical point, AB is a projection operator and thus recycling outputs is stable at the first pass: $(AB)^n(x) = AB(x) = U_{\mathcal{I}}U_{\mathcal{I}}^t(x)$ for any $n \geq 1$.

(8) Learning algorithms. Although a closed form expression is readily available for the global optimum, it is still worth studying the behavior of various learning algorithms, either

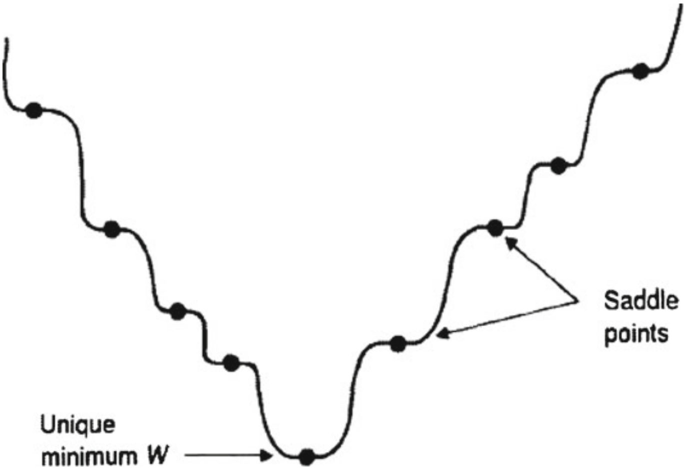

Fig. 3 Landscape of E in the linear real-valued and complex-valued case with squared Euclidean distance. All critical points are associated with projections onto subspaces spanned by the eigenvectors of the covariance matrix of $\mathcal{X}$. All critical points are saddle points, except those associated with the projection onto the subspace corresponding to the largest p eigenvalues

for comparison with cases where a closed form solution does not exist, or for potential embodiments that may not have direct access to the global minimum. Various descent algorithms can be used including gradient descent and alternate partial, or full, optimization of A and B, possibly combined with transposition (see [4] for details). These algorithms are convergent and generally converge to the global minimum when initialized randomly. In this linear case, alternate minimization of A and B can be viewed as an instance of the EM [8] algorithm under a standard Gaussian model.

(9) Generalization. At any critical point, for any x, $AB(x)$ is equal to the projection of x onto the corresponding subspace and the corresponding error can be expressed easily as the squared distance of x to the projection space.

(10) External targets. With the proper adjustments, the results above remain similar if a set of target output vectors $y_1, \ldots, y_m$ is provided, instead of $x_1, \ldots, x_m$ serving as the targets (see [3,4]).

(11) Composition. The global minimum of E remains the same if additional matrices of rank greater or equal to p are introduced between the input layer and the hidden layer or the hidden layer and the output layer. Thus there is no reduction in overall distortion by introducing such matrices. However, if such matrices are introduced for other reasons, there is a composition law so that the optimum solution for a deep autoencoder with a stack of matrices, can be obtained by combining the optimal solutions of each shallow autoencoders. More precisely, consider an autoencoder network with layers of size $n/p_1/p/p_1/n$ (Fig. 4) with $n > p_1 > p$. Then the optimal solution of this network can be obtained by first computing the optimal solution for an $n/p_1/n$ autoencoder network, and combining it with the optimal solution of an $p_1/p/p_1$ autoencoder network using the activities in the hidden layer of the first network as the training set for the second network, exactly as in the case of stacked RBMs [16,17]. This is because the projection onto the subspace spanned by the top p eigenvectors can be decomposed into a projection onto the subspace spanned by the top p_1 eigenvectors, followed by a projection onto the subspace spanned by the top p eigenvectors.

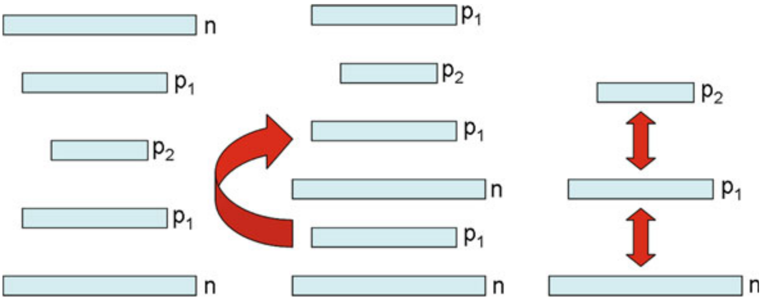

Fig. 4 Vertical composition of autoencoders (Color figure online)

4 The Boolean autoencoder

Boolean autoencoders correspond to the case where $\mathbb{F} = \mathbb{G} = \{0, 1\}$, $\mathcal{A}$ and $\mathcal{B}$ are classes of Boolean functions, and Δ is the Hamming distance. Traditionally a Boolean function is defined as a function from $\{0, 1\}^n$ to $\{0, 1\}$. 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 viewed as m ordinary Boolean functions. Again different interesting classes of Boolean autoencoders are derived by placing different restrictions on the classes of Boolean functions. Here we first study the unrestricted case, where $\mathcal{A}$ and $\mathcal{B}$ contain all possible Boolean functions of the right dimensions, and then the linear case where $\mathcal{A}$ and $\mathcal{B}$ are represented by matrices over GF(2). Another class, the Boolean threshold gate autoencoder, where the Boolean functions are restricted to being threshold gates, is considered in the Discussion.

The following definitions and notations will be useful in the statement of the main theorem. Given k binary column vectors $p_1, \ldots, p_k$ 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_{ji}. When n is even, there can be ties in which case one can flip a fair coin to assign the corresponding value.

Lemma 1 *The vector* $\text{Majority}(p)$ *is a vector in* $\mathbb{H}^n$ *closest to the center of gravity of the vectors* $p_1, \ldots, p_k$ *and it minimizes the function* $E(q) = \sum_1^k \Delta(q, p_i)$.

Proof The center of gravity is the vector c in $\mathbb{R}^n$ with coordinates $c_j = \left(\sum_{i=1}^k p_{ji}\right)/k$. For any j, $(p)_j$ is the closest binary value to $\mathbf{c}_j$. Furthermore:
$\sum_{i=1}^k \Delta(\text{Majority}(p), p_i) = \sum_{i=1}^k \sum_{j=1}^n \Delta(\text{Majority}(p)_j, p_{ji}) = \sum_{j=1}^n \left(\sum_{i=1}^k \Delta(\text{Majority}(p)_j, p_{ji})\right)$ 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's is a partition $\mathcal{C}^{Vor}(p_1), \ldots, \mathcal{C}^{Vor}(p_k)$ of $\mathbb{H}^n$ into k sets such that for any x in $\mathbb{H}^n$:

$$x \in \mathcal{C}^{Vor}(p_i) \Longrightarrow \forall j \quad \Delta(x, p_i) \leq \Delta(x, p_j) \tag{5}$$

Points that are equidistant from two or more p's can again be assigned arbitrarily to a unique center.

Theorem 1 (Boolean autoencoder) ***(1) Invariances***
(a) Permutations of hidden layer activities *The properties of the Boolean autoencoder are invariant under any one-to-one map from $\mathbb{H}^p$ to $\mathbb{H}^p$. Thus every solution is defined up to a permutation of the 2^p points of the hypercube $\mathbb{H}^p$.*
(b) Isometric change of coordinates in the input/output layers *Any isometry of the hypercube $\mathbb{H}^n$ preserving all the Hamming distances (hence also the Euclidean distances) between the points in $\mathcal{X}$ leads to an equivalent autoencoder problem.*
(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 $A^*(h_i) = \text{Majority}[\mathcal{X} \cap B^{-1}(h_i)]$.*
(3) Problem complexity *In general, the overall optimization problem is NP-hard. More precisely the optimization problem is NP-hard in the regime where $p \sim \epsilon \log_2 m$ with $\epsilon > 0$.*
(4) The landscape of E *In general E has many local minima (e.g with respect to the Hamming distance applied to the lookup tables of A and B).*
(5) Clustering *The overall optimization problem is a problem of optimal clustering. The clustering is defined by the transformation B.*

Proof (1) For the hidden layer corresponding to $\mathbb{H}^p$, the Boolean function are unrestricted and therefore their lookup tables can accommodate any such permutation, or relabeling of the hidden states. If C is a one-to-one map from $\mathbb{H}^p$ to $\mathbb{H}^p$, then $E(A, B) = E(AC, C-1B)$. For the input layer corresponding to $\mathbb{H}^n$, the property is obvious. Note that such isometries are generated by permutations and inversions of coordinates. (2) Assume first that A is fixed. Then for each of the 2^p possible Boolean vectors $h_1, \ldots, h_{2^p}$ of the hidden layer, $A(h_1) \ldots, A(h_{2^p})$ 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 $\mathcal{C}_1, \ldots, \mathcal{C}_{2^p}$, with $\mathcal{C}_i = \mathcal{C}^{Vor}(A(h_i))$. The optimal mapping B^* is then easily defined by setting $B^*(x) = h_i$ for any x in $\mathcal{C}_i = \mathcal{C}^{Vor}(A(h_i))$ (Fig. 5). Conversely, assume that B is fixed. Then for each of the 2^p possible Boolean vectors $h_1, \ldots, h_{2^p}$ 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 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 (Fig. 6). 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 $A^*(h_i$ should be the vector $\mathcal{X}$ that is closest to the vector $\text{Majority}[\mathcal{X} \cap B^{-1}(h_i)]$. (3) To be more precise, one must specify the regime of interest characterized by which variables among n, m, and p are going to infinity. Obviously one must have $n \to \infty$ and $m > 2^p$. If p does not go to infinity, then the problem can be polynomial, for instance when the centroids must belong to the training set. If $p \to \infty$ and m is a polynomial in n, which is the case of interest in machine learning where typically m is a low degree polynomial in n, then the problem of finding the best Boolean mapping (i.e. the Boolean mapping that minimizes the distortion E associated with the Hamming distance on the training set) is NP-hard, or the corresponding decision problem is NP-complete. In fact, the optimal clustering problem is NP hard when the number of clusters scales like $2^p \sim m^\epsilon$ ($\epsilon > 0$). The complexity proof is given in the next section. 4) Local minima can be defined with respect to small moves. In the Boolean case, we can define a critical point to be a point where both A and B are optimized, i.e. where $A = A^*(B)$ and $B = B^*(A)$. The distortion at such a point cannot be reduced by moving one point from

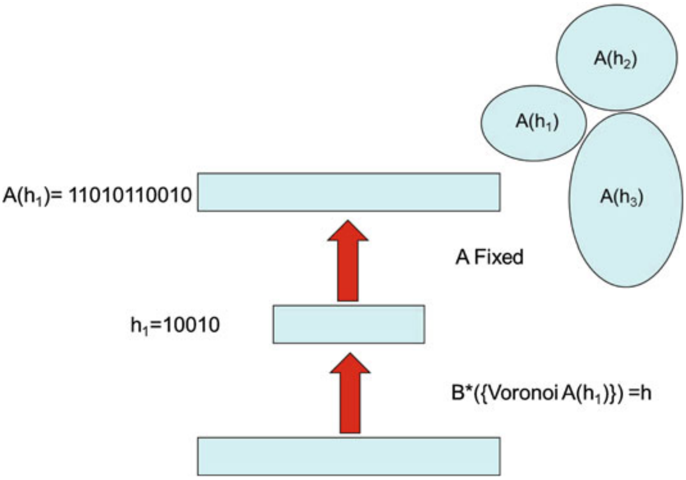

Fig. 5 Boolean autoencoder with fixed transformation A. For each vector h, in the hidden layer, A produces an output vector $A(h)$. The vectors $A(h)$ induce a Voronoi partition of the output space, hence of the input space. Any vector $x \in \mathbb{F}^n$ is in a partition of the form $A(h)$. In order to minimize the final distortion, one must have $B^*(x) = h$ (Color figure online)

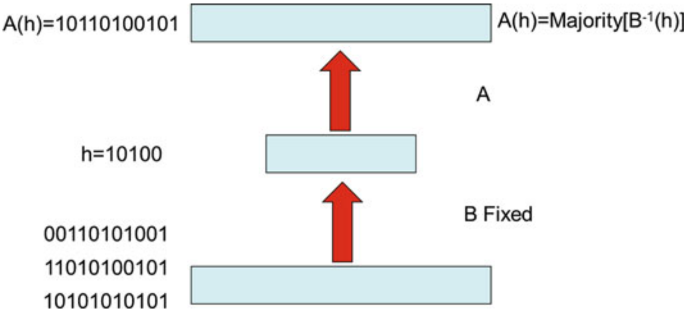

Fig. 6 Boolean autoencoder with fixed transformation B. As shown, consider the set of vectors $B^{-1}(h)$ in the input space that are mapped to the same vector vector h in the hidden layer. Then the vector $A(h)$ must have minimal average Hamming distance to all the vectors in $B^{-1}(h)$. This is achieved if $A(h)$ is the Majority vector of $\mathcal{X} \cap B^{-1}(h)$ for optimal performance on the training set, or $\cap B^{-1}(h)$ for optimal generalization (Color figure online)

its cluster to another cluster. Such critical points are local or global minima. The existence of local minima is not surprising since the optimization problem is NP-complete. Simple examples of local minima can be constructed (not shown). (5) This is obvious from the proof of (2). Note that approximate solutions can be sought by several algorithms, such as k-means, belief propagation [11], minimum spanning paths and trees [25], hierarchical clustering, and alternate optimization of A and B. Alternate optimization of A and B yields an approximate optimization algorithm for all autoencoders, since the distortion is positive and must decrease or stay constant at each optimization step. In the purely linear case over $\mathbb{R}$ or $\mathbb{C}$, or in the mixed case where the hidden layer is binary-valued but the output is real, the alternate optimization is closely related to the EM algorithm [8] and to the K-means algorithm [9], with the proper probabilistic interpretations.

5 The complexity of clustering on the hypercube

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. The complexity of various clustering problems, in different spaces, or with different objective functions, has been studied in the literature. There are primarily two kind of results: (1) graphical results derived on graphs $G = (V, E, \Delta)$ where the dissimilarity Δ is not necessarily a distance; and (2) geometric results derived in the Euclidean space $\mathbb{R}^d$ where $\Delta = L_2^2, L_2$, or L_1. In general, the clustering decision problem is NP-complete and the clustering optimization problem is NP-hard, except in some simple cases involving either a constant number k of clusters or clustering in the 1-dimensional Euclidean space. In general, the results in Euclidean spaces are harder to derive than the results on graphs. When polynomial time algorithms exist, geometric problems tend to have faster solutions taking advantage of the geometric properties. However, none of the existing complexity theorems directly addresses the problem of clustering on the hypercube with respect to the Hamming distance.

To deal with the hypercube clustering problem one must first understand which quantities are allowed to go to infinity. If n is not allowed to go to infinity, then the number m of training examples is also bounded by 2^n and, since we are assuming $p < n$, there is no quantity that can scale. Thus by necessity we must have $n \to \infty$. We must also have $m \to \infty$. The case of interest for machine learning is when m is a low degree polynomial of n. Obviously the hypercube clustering problem is in NP, and it is a special case of clustering in $\mathbb{R}^n$. Thus the only important problem to be addressed is the reduction of a known NP-complete problem to a hypercube clustering problem.

For the reduction, it is natural to start from a known NP-complete graphical or geometric clustering problem. In both case, one must find ways to embed the original problem with its original metric into the hypercube with the Hamming distance. There are theorems for homeomorphic or squashed-embedding of graphs into the hypercube [14,29], however these embeddings do not map the original dissimilarity function onto the the Hamming metric. Thus here we prefer to start from some of the known geometric results and use a strict cubical graph embedding. A graph is cubical if it is the subgraph of some hypercube $\mathbb{H}^d$ for some d [13,18]. Although deciding whether a graph is cubical is NP-complete [1], there is a theorem [15] 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; and (3) In each cycle of G, no color appears an odd number of times. We can now state and prove the following theorem.

Theorem 2 (Hypercube clustering) *Consider the problem:*
Problem *HYPERCUBE CLUSTERING*
Instance *m binary vectors $x_1, \ldots, x_m$ of length n and an integer k.*
Question *Can we find k binary vectors $c_1, \ldots, c_k$ of length n (the centroids) and a function f from $\{x_1, \ldots, x_m\}$ to $\{c_1, \ldots, c_k\}$ that minimizes the distortion $E = \sum_{t=1}^m \Delta(x_t, f(x_t)$ where Δ is the Hamming distance?*
The HYPERCUBE CLUSTERING problem is NP hard when $n \to \infty$ and $k \sim m^\epsilon$ ($\epsilon > 0$).

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 NP-complete [21] by reduction from 3-SAT [12] when $k \sim m^\epsilon$ ($\epsilon > 0$) (see, also related results in [19] and [28]). Without any loss

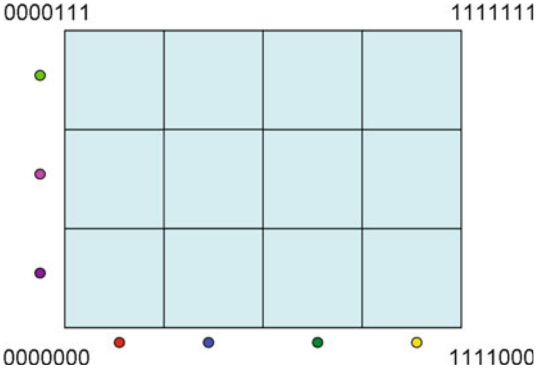

Fig. 7 Embedding of a 3 × 4 square lattice onto $\mathbb{H}^7$ by edge coloring. All edges in the *same row or column* are given the *same color*. *Each color* corresponds to one of the dimensions of the 7-dimensional hypercube. For any pair of points, their Manhattan distance on the lattice is equal to the Hamming distance between their images in the 7-dimensional hypercube (Color figure online)

of generality, we can assume that the points in these problems lie on the vertices of a square lattice. Using the theorem in [15], one can show that a $n \times m$ square lattice in the plane can be embedded into $\mathbb{H}^{n+m}$. In fact, an explicit embedding is given in Fig. 7. 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 $\mathbb{H}^{n+m}$. This polynomial reduction completes the proof that if the number of cluster satisfies $k = 2^p \sim m^\epsilon$, or equivalently $p \sim \epsilon \log_2 m \sim C \log_2 n$ (the latter when $m \sim n^C$), then the hypercube clustering problem associated with the Boolean autoencoder is NP-hard, and the corresponding decision problem 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.

6 Linear autoencoders over finite fields

Here we focus on linear autoencoders over GF(2), although we expect the results over other finite fields to be similar.

(1) Invariances.

(a) Change of coordinates in the hidden layer. For any linear invertible map C from $\mathbb{F}_2^p$ to $\mathbb{F}_2^p$, we have $E(A, B) = E(AC, C^{-1}B)$. Thus the problem is invariant under any linear change of coordinates in the hidden layer (this property is obviously true on any field).

(b) Change of coordinates in the input/output layers. The problem is invariant under isometric transformations in the input vectors, i.e transformations that preserve Hamming distances over the hypercube (and hence Euclidean distances too). One-to-one maps of the n-dimensional hypercube that preserve Hamming distances between points are generated by two kinds of transformations: (1) the $n!$ permutations of the coordinates, which in turn are generated by pairwise inversion of coordinates; and (2) the n maps corresponding to bit inversions for each component. Applying such transformations to the input vectors, generates new problems that are equivalent to the original problem.

(2) Fixed layer solutions. Clearly in this case there is no standard notion of convexity, even when A or B are fixed. Suppose for instance, that B is fixed. Then one is left with addressing a linear regression problem over GF(2) where the input vectors are the $B(x_i)'$s in $\mathbb{F}_2^p$ and the targets are the corresponding x_i's in $\mathbb{F}_2^n$ with respect to the Hamming distortion measure.

Clearly this leads to n separate regression problems, one for each output unit (or component). If we fix A, the situation is similar. Consider the 2^p possible Boolean vectors $\{h_1, \ldots, h_{2^p}\}$ in $h \in \mathbb{F}_2^p$ associated with the hidden layer, the corresponding images $\{A(h)_1), \ldots, A(h_{2^p})\}$ under A, and the Voronoi partition of $\mathbb{F}_2^n$ induced by these images (again breaking ties arbitrarily). For any input vector x_i, there is at least one vector h_k such that x_i is in the same Voronoi partition as $A(h_k)$. In other words, $\Delta(x_i, A(h_k)) \leq \Delta(x_i, A(h_l))$ for $l \neq k$, where Δ is the Hamming distance. In which case, we should have $B(x_i) = h_k$. Thus the optimal B is again defined by a regression problem with input-output pairs of the form (x_i, h_k) and the error function associated with Eq. 1, rather than the Hamming distance in $\mathbb{F}_2^p$. (Unlike the case where B is fixed, this does not correspond to p independent regression problems). Thus, in short, to make progress on this issue one must study the complexity of the linear regression problem over finite fields. The next sections show that linear regression over GF(2) and several other related decision problems are NP-complete.

7 The complexity of linear regression and the autoencoder problem over finite fields

7.1 Known preliminary results

We begin by stating five NP-complete problems that are closely related to the autoencoder problem over GF(2). Not surprisingly, these problems come from the coding literature.

Problem: MAXIMUM-LIKELIHOOD DECODING
Instance: A binary $m \times n$ matrix H, a vector $s \in \mathbb{F}_2^m$, and an integer $w > 0$.
Question: Is there a vector $x \in \mathbb{F}_2^n$ of weight $\leq w$, such that $Hx = s$?

The weight of a binary vector is defined as the number of 1-bits it contains. This problem was proved to be NP-complete using a reduction from THREE-DIMENSIONAL MATCHING [6]. From the original proof, the problem remains NP complete if the vector s is the vector of all 1s.

Problem: WEIGHT DISTRIBUTION
Instance: A binary $m \times n$ matrix H and an integer $w > 0$.
Question: Is there a vector $x \in \mathbb{F}_2^n$ of weight w, such that $Hx = 0$?

This problem was also proved to be NP-complete using a reduction from THREE-DIMENSIONAL MATCHING [6]. Note that any of these problems become solvable in polynomial time if either n or m is finite.

Problem: MINIMUM DISTANCE
Instance: A binary $m \times n$ matrix H and an integer $w > 0$.
Question: Is there a nonzero vector $x \in \mathbb{F}_2^n$ of weight $\leq w$, such that $Hx = 0$?

This decision problem was conjectured to be NP complete in [6] and was subsequently proved to be so [27]. The following two variants are also known to be NP-complete [22].

Problem: WEIGHT DISTRIBUTION (variant 1)
Instance: A binary $m \times n$ matrix H and an integer $w > 0$.
Question: Is there a vector $x \in \mathbb{F}_2^n$ of weight $\geq w$, such that $Hx = 0$?

Problem: WEIGHT DISTRIBUTION (variant 2)
Instance: A binary $m \times n$ matrix H, integers $w_2 > w_1 > 0$.
Question: Is there a vector $x \in \mathbb{F}_2^n$ such that $Hx = 0$ and $w_1 \leq wt(x) \leq w_2$?

7.2 Kernel, image, and subspace problems

The WEIGHT DISTRIBUTION, MINIMUM DISTANCE, WEIGHT DISTRIBUTION (variant 1), and WEIGHT DISTRIBUTION (variant 2) can equivalently be viewed as problems regarding the weight of vectors in the kernel of H. Thus the equivalent kernel problems of finding a non-zero vector in the kernel of H of weight equal to w, less or equal to w (minimal weight), greater or equal to w (maximal weight), or in the range $[w_1, w_2]$, are all NP-complete, yielding the following theorem.

Theorem 3 (Kernel over GF(2))
Given a binary $m \times n$ matrix H, the problems of determining whether the kernel of H contains a non-zero vector with the corresponding properties:
KERNEL OVER GF(2) ($= w$)
KERNEL OVER GF(2) ($\leq w$)
KERNEL OVER GF(2) ($\geq w$)
KERNEL OVER GF(2) ($w_1 \leq w \leq w_2$)
are all NP-complete.

As noted in [6,27], these problems remain NP-complete when the rows of H are independent, i.e. when H has rank m and $m \leq n$. These problems remain NP-complete if one of the components of x is constrained to be 1. To see this, it is first obvious to note that they remain in the class NP. Furthermore any unconstrained kernel problem defined by an $m \times n$ matrix H can be reduced to a constrained problem wit an $m \times (n + 1)$ matrix, by adding a column of 0s to the matrix H, and adding a corresponding component fixed to 1 to the vector x. This implies immediately that the four-forms of MAXIMUM-LIKELIHOOD DECODING of finding a vector x of of weight equal to w, less or equal to w (minimal weight), greater or equal to w (maximal weight), or in the range $[w_1, w_2]$, are all NP-complete, yielding the following theorem.

Theorem 4 (General maximum likelihood decoding) *Given a binary $m \times n$ matrix H and a vector $s \in \mathbb{F}_2^m$, the problems of determining whether there exists a vector x satisfying $Hx = s$ with the corresponding properties:*
MAXIMUM-LIKELIHOOD DECODING ($= w$)
MAXIMUM-LIKELIHOOD DECODING ($\leq w$)
MAXIMUM-LIKELIHOOD DECODING ($\geq w$)
MAXIMUM-LIKELIHOOD DECODING ($w_1 \leq w \leq w_2$)
are all NP-complete.

For any linear operator H, any vector $x \in \mathbb{F}_2^n$ can be decomposed uniquely as $x = u + v$ with $u \in KerH$ and $v \in J$, where $KerH$ is the kernel of H and J is a subspace of $\mathbb{F}_2^n$ isomorphic to the image ImH. Furthermore, it is easy to find a basis of $KerH$ or J in polynomial time using standard algebraic manipulations and create a new matrix G such that $ImG = KerH$ (for instance take the $n \times n$ matrix whose columns form a basis of H, padded with 0s as needed) or $KerG = ImH$. Thus the equivalent image problems of finding a non-zero vector in the image of H of weight equal to w, less or equal to w (minimal weight), greater or equal to w (maximal weight), or in the range $[w_1, w_2]$, are all NP-complete yielding the following theorem.

Theorem 5 (Image over GF(2)) *Given a binary $m \times n$ matrix H, the problems of determining whether the image of H contains a non-zero vector with the corresponding properties:*

IMAGE OVER GF(2) (= w)
IMAGE OVER GF(2) ($\leq w$)
IMAGE OVER GF(2) ($\geq w$)
IMAGE OVER GF(2) ($w_1 \leq w \leq w_2$)
are all NP-complete.

More generally, any subspace can be viewed as the image of a matrix H comprising a basis of the subspace as its column vector. Thus the subspace problems of finding a non-zero vector in a subspace of dimension n of $\mathbb{F}_2^m$ of weight equal to w, less or equal to w (minimal weight), greater or equal to w (maximal weight), or in the range $[w_1, w_2]$, are all NP-complete yielding the following theorem.

Theorem 6 (Subspace over GF(2)) *Given a binary $m \times n$ matrix H, the problems of determining whether the subspace generated by the columns of H contains a non-zero vector with the corresponding properties:*
SUBSPACE OVER GF(2) (= w)
SUBSPACE OVER GF(2) ($\leq w$)
SUBSPACE OVER GF(2) ($\geq w$)
SUBSPACE OVER GF(2) ($w_1 \leq w \leq w_2$)
are all NP-complete.

7.3 Linear regression

We can now consider the general problem of linear regression over GF(2). We first consider exact (i.e. with no distortion) linear regression, and then approximate linear regression where the goal is to minimize the distortion.

Problem: EXACT LINEAR REGRESSION OVER GF(2)
Instance: Two integers m and n, an $m \times n$ binary matrix H, an integer w, and a vector $y \in \mathbb{F}_2^m$.
Question: Is there a vector $x \in \mathbb{F}_2^n$ of weight $\leq w$ such that $Hx = y$?

Note that in this notation the m rows of H are the input vectors, and the components of y are the corresponding targets. This decision problem is of course NP-complete since it is the same problem as MAXIMUM-LIKELIHOOD DECODING viewed from a linear regression perspective, where the rows of the matrix H are the input vectors and the components of y are the corresponding targets. Note that the same decision problem without the restriction $x \leq w$ can be solved in polynomial time using standard algebraic manipulations to solve the corresponding linear system. Thus the following version of the problem, where the vectors x is required to have weight equal to w, less or equal to w (minimal weight), greater or equal to w (maximal weight), or in the range $[w_1, w_2]$, are all NP-complete yielding the following theorem.

Theorem 7 (Exact linear regression over GF(2)) *Given an $m \times n$ binary matrix H and a vector $y \in \mathbb{F}_2^m$, the problems of finding a vector x satisfying $Hx = y$ with the corresponding properties:*
EXACT LINEAR REGRESSION (= w)
EXACT LINEAR REGRESSION ($\leq w$)
EXACT LINEAR REGRESSION ($\geq w$)
EXACT LINEAR REGRESSION ($w_1 \leq w \leq w_2$)
are all NP-complete.

We now turn to approximate linear regression where we measure the distortion $\Delta(Hx, y)$ with the Hamming distance Δ. For any two binary vectors u and v, note that $\Delta(u, v) = wt(u+v)$ since $u_i + v_i = 0$ if and only if $u_i = v_i$. We have the following result.

Theorem 8 (Approximate linear regression over GF(2)) *The decision problem*
Problem *APPROXIMATE LINEAR REGRESSION OVER GF(2)*
Instance *Two integers m and n, an $m \times n$ binary matrix H, an integer w, and a vector $y \in \mathbb{F}_2^m$.*
Question *Is there a vector $x \in \mathbb{F}_2^n$ of such that $\Delta(Hx, y) = wt(Hx + y) \leq w$?*
is NP-complete.

Proof To see this, we first construct a new $m \times (n+1)$ binary matrix H' by appending y to H. Consider the vector x' obtained by obtaining appending a 1 to x. Then $Hx + y = H'x'$ thus the problem becomes a MAXIMUM-LIKELIHOOD DECODING problem with the matrix H', the target y, and the last component of the vector $x'1$ constrained to 1.

By the results above on MAXIMUM-LIKELIHOOD DECODING, the linear regression problems over GF(2) with distortion equal to w, less or equal to w (minimal weight), greater or equal to w (maximal weight), or in the range $[w_1, w_2]$, are all NP-complete yielding the following theorem.

Theorem 9 (Approximate linear regression over GF(2)) *Given an $m \times n$ binary matrix H and a vector $y \in \mathbb{F}_2^m$, the problems of finding a vector x so that the distortion $\Delta(Hx, y)$ satisfies the corresponding properties:*
LINEAR REGRESSION ($= w$)
LINEAR REGRESSION ($\leq w$)
LINEAR REGRESSION ($\geq w$)
LINEAR REGRESSION ($w_1 \leq w \leq w_2$)
are all NP-complete.

7.4 Linear autoencoders over GF(2)

We can now address the complexity of linear autoencoders over GF(2).

Problem: LINEAR AUTOENCODER OVER GF(2)
Instance: Four integers n, p, m, and w. A set of m vectors $x_1, \ldots, x_m$ in $\mathbb{F}_2^n$.
Question: Is there an $p \times n$ matrix B and an $n \times p$ matrix A such that $\sum_{i=1}^m wt(ABx_i + x_i) \leq w$?
Note again that $wt(ABx + x)$ is simply the Hamming distance between ABx and x. By taking $m = n$ the problem is at least as hard as LINEAR REGRESSION and therefore it is NP-complete. More generally, by the same argument, we see that the LINEAR AUTOENCODER OVER GF(2) with distortion equal to w, less or equal to w (minimal weight), greater or equal to w (maximal weight), or in the range $[w_1, w_2]$, are all NP-complete yielding the following theorem.

Theorem 10 (Linel autoencodes over GF(2)) *Given m binary vectors $x_1, \ldots, x_m$ over $\mathbb{F}_2^n$, the problems of finding an $n \times p$ matrix A and an $p \times n$ matrix B so that the overall distortion $\sum_i \Delta(ABx_i, x_i)$ satisfies the corresponding properties:*
LINEAR AUTOENCODER ($= w$)
LINEAR AUTOENCODER ($\leq w$)
LINEAR AUTOENCODER ($\geq w$)
LINEAR AUTOENCODER ($w_1 \leq w \leq w_2$)
are all NP-complete.

The problem remains NP-complete if one adds the constraint $p < n$.
It is reasonable to conjecture that all the KERNEL, IMAGE, SUBSPACE, EXACT LINEAR REGRESSION, LINEAR REGRESSION, and AUTOENCODER problems described above remain NP complete on finite fields other than GF(2).
Some of the results above can be restated almost immediately in terms of set intersections or parity gates.

7.5 Complexity of set intersections

This appendix shows that some of the problems examined in the section on linear autoencoders over finite fields can be restated in terms of set intersections or parity gates.

Problem: EVEN SET INTERSECTIONS ($\leq w$)
Instance: A collection C of m subsets of a set S of size n, and an integer $w > 0$.
Question: Is there a non-empty subset of S of size $\leq w$ that has an even intersection with all the members of C?

Obviously this is equivalent to MINIMUM DISTANCE OR KERNEL OVER GF(2) ($\leq w$). Thus, using the obvious notation, we have the following theorem.
begintheorem(Even set intersections) Given a collection C of m subsets of a set S of size n. The problems of finding a subset of S with an even non-zero intersection with all the subsets in C and the corresponding size properties:
EVEN SET INTERSECTIONS ($= w$)
EVEN SET INTERSECTIONS ($\leq w$)
EVEN SET INTERSECTIONS ($\geq w$)
EVEN SET INTERSECTIONS ($w_1 \leq w \leq w_2$)
are all NP-complete.
The results are similar for odd intersections. In fact, consider the following problem.

Problem: ODD SET INTERSECTIONS ($\leq w$)
Instance: A collection C of m subsets of a set S of size n, and an integer $w > 0$.
Question: Is there a subset of S of size $\leq w$ that has an odd intersection with all the members of C?

This is exactly equivalent to MAXIMUM-LIKELIHOOD DECODING with s being the vector of all 1s, and similarly for the other versions, yielding the following theorem.

Theorem 11 (Odd set intersections) *Given a collection C of m subsets of a set S of size n. The problems of finding a subset of S with an odd intersection with all the subsets in C and the corresponding size properties:*
ODD SET INTERSECTIONS ($= w$)
ODD SET INTERSECTIONS ($\leq w$)
ODD SET INTERSECTIONS ($\geq w$)
ODD SET INTERSECTIONS ($w_1 \leq w \leq w_2$)
are all NP-complete.

7.6 Complexity of parity gates

Some of these problems can also be restated in terms of parity gates. A parity gate or parity Boolean function f with binary inputs $u_1, \ldots, u_n$ and support $u_{i_1}, \ldots, u_{i_{i_k}}$ can be defined as the Boolean function that produces a 0 when the number of 1-bits in its support is even, and 0 otherwise. Thus f can be seen as a linear operator in $\mathbb{F}_2^n$ associated with the vector whose components are 1 on the support and 0 otherwise. By adding an input x_0 which is always equal to 1, it is easy to construct a gate f' that behaves like the negation of f, i.e. produces a 1 if the number of 1-bits in the support of f is odd. For simplicity, we call such a gate also a parity gate. Thus the problem of finding a parity function implementing a set of input output relationships exactly or approximately with respect to the Hamming distance with a support of size equal to w, less or equal to w (minimal weight), greater or equal to w (maximal weight), or in the range $[w_1, w_2]$, are all NP-complete yielding the following theorems.

Theorem 12 (Exact parity gates) *Given m vectors $h_1, \ldots, h_m$ in $\mathbb{F}_2^n$ and a vector y in $\mathbb{F}_2^m$, the problems of finding a parity gate f such that $f(h_i) = y_i$ for $i = 1, \ldots, m$ with support size satisfying the corresponding properties*
EXACT PARITY GATES $(= w)$
EXACT PARITY GATES $(\leq w)$
EXACT PARITY GATES$(\geq w)$
EXACT PARITY GATES $(w_1 \leq w \leq w_2)$
are all NP-complete.

Theorem 13 (Parity gates) *Given m vectors $h_1, \ldots, h_m$ in $\mathbb{F}_2^n$ and a vector y in $\mathbb{F}_2^m$, the problems of finding a parity gate with the total distortion $\sum_{i=1}^m \Delta(f(h_i, y_i)$ satisfying the properties*
PARITY GATES $(= w)$
PARITY GATES $(\leq w)$
PARITY GATES$(\geq w)$
PARITY GATES $(w_1 \leq w \leq w_2)$
are all NP-complete.

8 Discussion and conclusion

Here we provide a brief discussion of other autoencoders and autoencoder regimes which can either be folded under the previous analyses or are beyond the scope and space of this article.

8.1 Mixed autoencoders

First, one can consider mixed autoencoders with different constraints on $\mathbb{F}$ and $\mathbb{G}$, or different constraints on $\mathcal{A}$ and $\mathcal{B}$. A simple example is when the input and output layers are real $\mathbb{F} = \mathbb{R}$ and the hidden layer is binary $\mathbb{G} = \{0, 1\}$ (and $\Delta = L_2^2$). It is easy to check that in this case, as long as $2^p = k < m$, the autoencoder aims at clustering the real data into k clusters and all the results obtained in the Boolean case are applicable with the proper adjustments. For instance, the centroid associated with a hidden state h should be the center of mass of the input vectors mapped onto h. In general, the optimization decision problem for these autoencoders is also NP-complete and, more importantly, from a probabilistic view point, they correspond exactly to a mixture of k Gaussians model. Other interesting mixed examples can be derived when $\mathcal{A}$ and $\mathcal{B}$ correspond to different classes of Boolean functions, such as: (1) unrestricted; (2) linear over GF(2); (3) threshold gates; and (4) monotone functions, in all possible combinations.

8.2 Threshold gate autoencoders

Threshold gate autoencoders are a special class of Boolean autoencoders where all the Boolean functions are threshold gate functions. A threshold gate function defined on n inputs $u_1, \ldots u_n$ with weights $w_1, \ldots, w_n$ and threshold t is the function $f(u_1, \ldots, u_n) = 1$ if $\sum_{i=1}^n w_i u_i > t$, and $f(u_1, \ldots, u_n) = 0$ otherwise. These functions are of interest because they can be viewed as the limiting case of standard sigmoidal neural network functions when the gain of the sigmoidal functions is taken to infinity. They are also closely related to Support Vector Machines [24].

We conjecture that the Threshold Gate Autoencoder decision problem is NP-complete. This is a reasonable conjecture based on the result that training a 3-node neural network is NP-compete [7]. This result is obtained by showing that training a neural network of threshold gates with n inputs, two hidden nodes, and one output node is NP-complete by reduction of the SET SPLITTING (or HYPERGRAPH 2-COLORABILITY) problem. In fact, this problem can be embedded into a threshold gate autoencoder problem as follows. Consider an autoencoder of threshold gates with $n + 1$ inputs, $n + 2$ hidden units, and $n + 1$ outputs. The $n + 1$ inputs correspond to the same $x_1, \ldots, x_n$ inputs of [7] augmented with the corresponding target value $x_{n+1} = t$ taken from [7]. The inputs $x_1, \ldots, x_n$ are fully connected to the hidden units $h_1, \ldots, h_n$, which in turn are fully connected to the outputs $z_1, \ldots, z_n$ (allowing for an easy implementation of the identity function). The inputs $x_1, \ldots, x_n$ are also fully connected to the two hidden units h_{n+1}, h_{n+2}, which in turn are connected to the output z_{n+1} thereby implementing the circuit used in [7]. These two hidden units are aslo fully connected to the outputs $z_1, \ldots, z_n$. The input $x_{n+1} = y$ is fully connected to the hidden units $h_1, \ldots, h_n$. Thus the problem used in [7] becomes a sub-problem of a threshold gate autoencoder problem, where some of the connections are missing or forced to be 0. There are $n + 2$ connections forced to be 0 in the construction above to avoid any communication between x_{n+1} and z_{n+1}. Thus this construction proves the following slightly weaker theorem.

Theorem 14 (Threshold gate autoencoder) *The threshold gate autoencoder decision problem where up to a linear number of connections are set to 0 is NP-complete.*

Although the threshold gate autoencoder optimization problem may be NP-hard, there are interesting optimization strategies that can be applied to it, in particular alternate optimization. This is because for fixed A (resp. fixed B, we know from the theory of the unrestricted Boolean autoencoder developed here what is the optimal Boolean function that B (resp. A) ought to implement. Thus we know what the inputs and outputs of such function ought to be and therefore we are reduced to the problem of training a single layer threshold gate network, or a perceptron, using known input and output targets. When the data is linearly separable, this can be solved using the well-known perceptron algorithm. When the data is not linearly separable, maximum margin approximate solutions can be found using standard Support Vector Machine algorithms [24]. Thus in any case, there are polynomial time efficient algorithms for optimizing B (resp. A) individually, and proceeding with alternate optimization. This yields a novel approximate algorithm for training threshold gate autoencoders. Furthermore, the algorithm can immediately be extended to autoencoders with multiple layers, as well as to the non-associative case, providing a novel algorithm for training multilayer networks of threshold gates, or other functions, with pre-defined inputs and outputs. Basically the idea of the algorithm is to train one layer or fraction of a layer at a time, using the analyses above to provide suitable output

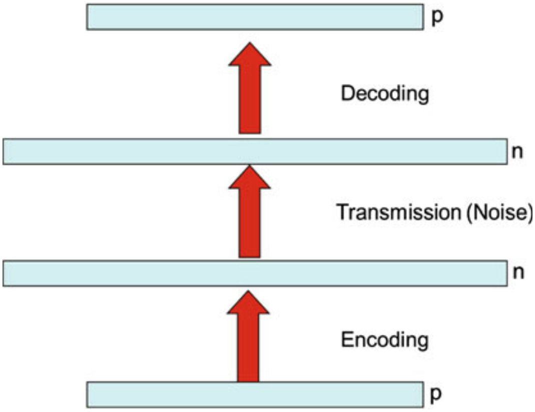

Fig. 8 Standard channel coding framework (Color figure online)

targets for training. Simulations of this approach are in progress and will be presented elsewhere.

8.3 Autoencoders with $p \geq n$

When the hidden layer is larger than the input layer and $\mathbb{F} = \mathbb{G}$, there is an optimal 0-distortion solution involving the identity function. Thus this case is interesting only if additional constraints are added to the problem. These can come in many forms, for instance in terms of restrictions on the classes of functions $\mathcal{A}$ and $\mathcal{B}$ or in terms of regularization, for instance to ensure sparsity of the hidden-layer representation. When these constraints force the hidden layer to assume only k different values and $k < m$, for instance in the case of a sparse Boolean hidden layer, then some of the previous analyses hold and the problem reduces to a k clustering problem. Two important concept that can lead to large hidden layer are the horizontal composition of autoencoders [2], and the presence of "communication" noise as discussed below.

8.4 Autoencoders, coding, and information theory

Not surprisingly, autoencoders are closely related to the theory of information and coding [20]. The autoencoders studied in this paper ($p < n$) are implementing a form of compression, which in general is bound to be lossy if the entropy of the input data exceeds the entropy of what can be represented in the hidden layer. Linear autoencoders over GF(2) can be viewed as linear codes for compression. In the dual case of communication over noisy channels (or storage in noisy media), in general messages are expanded before their transmission (or storage), with encoding schemes that include for instance parity bits, and then decoded at the receiving end. This case corresponds to autoencoders with expansive hidden layers ($p > n$) (Fig. 8), with the linear case encompassing standard linear codes for communication.

8.5 Open questions

In closing, autoencoders have a rich mathematical structure and lead to several additional problems that have not been addressed here. Examples of such problems include: (1) the theory of linear autoencoders over finite fields other than GF(2); (2) the derivation of exact and efficient algorithms for the unrestricted Boolean autoencoder, or the linear autoencoder over GF(2), when the number of clusters is fixed (p fixed); (3) the derivation of efficient approximate algorithms for all the NP-hard problems described here; and (4) the study of the other properties of linear autoencoders over GF(2) (e.g. transposition); and (5) the study of the family of learning algorithms for deep architectures briefly mentioned above.

Acknowledgments Work supported in part by grants NSF IIS-0513376, NIH LM010235, and NIH NLM T15 LM07443 to PB.

References

1. Afrati F., Papadimitriou C., Papageorgiou G.: The complexity of cubical graphs. Inf. control **66**(1–2), 53–60 (1985).
2. Baldi P.: Autoencoders, unsupervised learning, and deep architectures. In: Journal of Machine Learning Research, Workshop and Conference Proceedings, Proceedings of the 2011 ICML Workshop on Unsupervised and Transfer Learning, vol. 27, Bellevue, WA, pp. 37–50 (2012).
3. Baldi P., Hornik K.: Neural networks and principal component analysis: learning from examples without local minima. Neural Netw. **2**(1), 53–58 (1988).
4. Baldi P., Lu Z.: Complex-valued autoencoders. Neural Netw. **33**, 136–147 (2012).
5. Bengio Y., LeCun Y.: Scaling learning algorithms towards AI. In: Bottou L., Chapelle O., DeCoste D., Weston J. (eds.) Large-scale kernel machines. MIT Press, Cambridge (2007).
6. Berlekamp E., McEliece R., van Tilborg H.: On the inherent intractability of certain coding problems(Corresp.). IEEE Trans. Inf. Theory **24**(3), 384–386 (1978).
7. Blum A., Rivest R.: Training a 3-node neural network is np-complete. Neural Netw. **5**(1), 117–127 (1992).
8. Dempster A.P., Laird N.M., Rubin D.B.: Maximum likelihood from incomplete data via the em algorithm. J. R. Stat. Soc. **B39**, 1–22 (1977).
9. Duda R.O., Hart P.E., Stork D.G.: Pattern classification, 2nd edn. Wiley, New York (2000).
10. Erhan D., Bengio Y., Courville A., Manzagol P.A., Vincent P., Bengio S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11**, 625–660 (2010).
11. Frey B., Dueck D.: Clustering by passing messages between data points. Science **315**(5814), 972 (2007).
12. Garey M., Johnson D.: Computers and intractability. Freeman, San Francisco (1979).
13. Harary F.: Cubical graphs and cubical dimensions. Comput. Math. Appl. **15**(4), 271–275 (1988).
14. Hartman J.: The homeomorphic embedding of Kn in the m-cube* 1. Discret. Math. **16**(2), 157–160 (1976).
15. Havel I., Morávek J.: B-valuations of graphs. Czechoslov. Math. J. **22**(2), 338–351 (1972).
16. Hinton G., Osindero S., Teh Y.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006).
17. Hinton G., Salakhutdinov R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504 (2006).
18. Livingston M., Stout Q.: Embeddings in hypercubes. Math. Comput. Model. **11**, 222–227 (1988).
19. Mahajan M., Nimbhorkar P., Varadarajan K.: The planar k-means problem is NP-hard. In: Proceedings of 3rd Annual Workshop on Algorithms and Computation WALCOM, Kolkata, pp. 274–285 (2009).
20. McEliece R.J.: The theory of information and coding. Addison-Wesley Publishing Company, Reading (1977).
21. Megiddo N., Supowit K.: On the complexity of some common geometric location problems. SIAM J. Comput. **13**(1), 182–196 (1984).
22. Ntafos S., Hakimi S.: On the complexity of some coding problems (corresp.). IEEE Trans. Inf. Theory **27**(6), 794–796 (1981).
23. Rumelhart D., Hinton G., Williams R.: Learning internal representations by error propagation. In: Parallel distributed processing, vol 1: Foundations. MIT Press, Cambridge (1986).
24. Scholkopf, B., Smola, A.J.: Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge (2002).
25. Slagle J., Chang C., Heller, S.: A clustering and data reorganization algorithm. IEEE Trans. Syst. Man Cybern. **5**, 121–128 (1975).

26. Sutskever I., Hinton G.: Deep, narrow sigmoid belief networks are universal approximators. Neural Comput. **20**(11), 2629–2636 (2008).
27. Vardy A.: The intractability of computing the minimum distance of a code. IEEE Trans. Inf. Theory **43**(6), 1757–1766 (1997).
28. Vattani A.: A simpler proof of the hardness of k-means clustering in the plane. UCSD Technical Report (2010).
29. Winkler P.: Proof of the squashed cube conjecture. Combinatorica **3**(1), 135–139 (1983).