Deep Belief Networks are compact universal approximators

Nicolas Le Roux[1], **Yoshua Bengio**[2]

[1]Microsoft Research Cambridge

[2]University of Montreal

Keywords: Deep Belief Networks, Universal Approximation

Abstract

Deep Belief Networks (DBN) are generative models with many layers of hidden causal variables, recently introduced by Hinton *et al.* (2006), along with a greedy layer-wise unsupervised learning algorithm. Building on Le Roux and Bengio (2008) and Sutskever and Hinton (2008), we show that deep but narrow generative networks do not require more parameters than shallow ones to achieve universal approximation. Exploiting the proof technique, we prove that deep but narrow feed-forward neural networks with sigmoidal units can represent any Boolean expression.

1 Introduction

Sigmoidal belief networks are generative probabilistic models with hidden variables organized in layers, with each hidden variable conditionally binomial given the values of variables in the layer above, and the conditional probability taking the form of a traditional sigmoidal neuron Neal (1992). Upper layers represent more "abstract" con-

cepts that explain the input observation $\mathbf{x}$, whereas lower layers are expected to extract "low-level features" from $\mathbf{x}$.

Deep Belief Networks (DBN) Hinton *et al.* (2006) are particular sigmoidal belief networks for which a clever and successful learning algorithm has been proposed, using as a building block a Restricted Boltzmann Machine (RBM) Smolensky (1986); Freund and Haussler (1991), representing one layer of the model. Although computing the exact log-likelihood gradient of an RBM is intractable, RBMs have been successfully trained using estimators of the log-likelihood gradient such as the Contrastive Divergence algorithm Hinton (2002); Hinton *et al.* (2006) and the persistent Contrastive Divergence algorithm Tieleman (2008). The algorithm proposed in Hinton *et al.* (2006) to train a DBN is a greedy layer-wise training algorithm in which the k-th layer is first trained as an RBM modeling the output (samples from the posterior) of layer $k-1$. This greedy layer-wise strategy has been found to work for other similar unsupervised models for deep architectures Bengio (2009), based on improved auto-encoder variants Bengio *et al.* (2007); Ranzato *et al.* (2007); Vincent *et al.* (2008). One of the motivations for deep architectures is that they can sometimes be exponentially more efficient than shallow ones in terms of number of elements needed to represent a function. More precisely, there are functions that can be represented compactly with a neural network of depth k but that would require exponential size (with respect to input size) networks of depth $k-1$ Hastad and Goldmann (1991); Bengio (2009). This begs the question: can we have guarantees of the representational abilities of models with potentially deep architectures such as DBNs? Sutskever and Hinton (2008) showed that a deep but narrow DBN (with only $n+1$ units per layer and n binary inputs) can represent any distribution on its input, with no more than 3×2^n layers. Unfortunately, this makes the number of parameters on the order of $3(n+1)^2 2^n$, larger than the number of parameters required by a single level but very fat DBN (i.e. an RBM), which is on the order of $n2^n$. This is the starting point for this paper.

The main results are the following. If n is a power of 2, i.e. $n = 2^t$, then a Deep Belief Network composed of $\frac{2^n}{n} + 1$ layers of size n is a universal approximator for distributions over $\{0,1\}^n$. This improves on the 3×2^n upper bound on the number of layers already shown in Sutskever and Hinton (2008), and makes the number of parameters of a DBN universal approximator no worse than the number of parameters

of a single RBM. It remains to be shown whether this is also a lower bound on the number of parameters required to achieve universal approximation. If it were true it would imply (as we expected) that the large efficiency gains one can potentially obtain with DBNs are not universal but only obtainable for some specific target functions.

Using the same technique, the paper also shows that a deep but narrow feedforward deterministic neural network can represent any function from $\{0,1\}^n$ to $\{0,1\}$, a slight improvement over the result proved in Rojas (2003).

2 Deep Belief Nets

In this section we briefly review the Deep Belief Net (DBN) model as proposed in Hinton *et al.* (2006), introducing notation for the rest of the paper.

Let $\mathbf{h}^i$ represent the vector of hidden variables at layer i. The model is parametrized as follows:

$$P(\mathbf{x}, \mathbf{h}^1, \mathbf{h}^2, \cdots \mathbf{h}^\ell) = P(\mathbf{x}|\mathbf{h}^1)P(\mathbf{h}^1|\mathbf{h}^2)\cdots P(\mathbf{h}^{\ell-2}|\mathbf{h}^{\ell-1})P(\mathbf{h}^\ell, \mathbf{h}^{\ell-1})$$

where all the conditional layers $P(\mathbf{h}^i|\mathbf{h}^{i+1})$ are factorized conditional distributions for which the computation of probability and sampling are very easy. In Hinton *et al.* (2006) one considers the hidden layer $\mathbf{h}^i$ a binary random vector with elements $\mathbf{h}^i_j$ and

$$P(\mathbf{h}^i|\mathbf{h}^{i+1}) = \prod_{j=1}^{n_i} P(\mathbf{h}^i_j|\mathbf{h}^{i+1}) \tag{1}$$

with element $\mathbf{h}^i_j$ a *stochastic neuron* or **unit**, whose binary activation is 1 with probability

$$P(\mathbf{h}^i_j = 1|\mathbf{h}^{i+1}) = \mathrm{sigm}\left(b^i_j + \sum_{k=1}^{n_{i+1}} W^i_{jk}\mathbf{h}^{i+1}_k\right) \tag{2}$$

where $\mathrm{sigm}(a) = 1/(1+\exp(-a))$ is the usual sigmoidal activation function, the b^i_j are called the *biases* (for unit j of layer i) and W^i is called the *weight matrix* for layer i. If we denote $\mathbf{h}^0 = \mathbf{x}$, the generative model for the first layer $P(\mathbf{x}|\mathbf{h}^1)$ also follows eq. 1 and 2. The joint distribution $P(\mathbf{h}^\ell, \mathbf{h}^{\ell-1})$ of the top two layers is a Restricted Boltzmann Machine (RBM), described in Hinton *et al.* (2006). A DBN is thus a particular kind of sigmoidal belief network where the top level prior comes from an RBM.

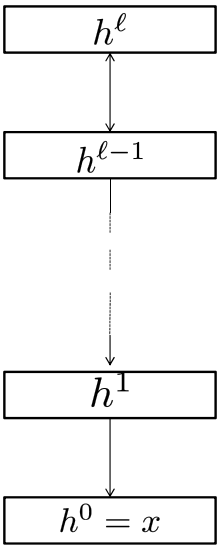

Figure 1: Graphical representation of a Deep Belief Network. The connection between the top layers is undirected and the connections between all the lower layers are directed.

3 Gray Code Deep Belief Network

In this section we use one or more Gray code sequences (see section 3.2 for a definition of such sequences) in order to capture arbitrary discrete distributions with a DBN. This is inspired by the work of Sutskever and Hinton (2008) in which, by adding layers, one constructively changes the probability for one of the 2^n configurations, so as to produce the desired distribution. To avoid confusion, we used different terms for probabilistic and deterministic changes of bits:

- "switch" refers to a deterministic change from 0 to 1 (or from 1 to 0)
- "flip" refers to a probabilistic change from 0 to 1 (or from 1 to 0).

Therefore, we shall say "bit k is switched from 0 to 1" and "bit k is flipped from 0 to 1 with probability p".

3.1 Overview

Let us assume we are given an arbitrary target distribution p^* over binary vectors of size n, which we want to capture with a DBN. The method proposed in Sutskever and Hinton (2008) is the following:

- Define an arbitrary sequence $(a_i)_{1 \leq i \leq 2^n}$ of binary vectors in $\{0,1\}^n$.
- Let the top-level RBM (between layers $\mathbf{h}^{3 \cdot 2^n - 2}$ and $\mathbf{h}^{3 \cdot 2^n - 3}$) assign probability 1 to a_1

- Using a specific sigmoid belief network composed of three layers, generate a_2 with probability $1 - p^*(a_1)$ and a_1 with probability $p^*(a_1)$ (specific details on the architecture of such a network are to be found in their paper), yielding the correct probability for a_1. We shall refer to this operation as a "transfer" of probability mass from a_1 to a_2.

- The subsequent 3-layer sigmoid belief network acts as follows:

    1. If the vector on $\mathbf{h}^{3 \cdot 2^n - 6}$ is a_2, transfer $1 - p^*(a_1) - p^*(a_2)$ of its probability mass to a_3.
    2. Otherwise, copy the vector on $\mathbf{h}^{3 \cdot 2^n - 6}$ to $\mathbf{h}^{3 \cdot 2^n - 9}$

- Continue in the same way, transferring each time probability mass from a_k to a_{k+1} vectors while leaving the other vectors unchanged.

At the end of this procedure, all the mass has been appropriately transferred to the $(a_i)_{2 \leq i \leq 2^n}$ and the desired distribution p^* is obtained.

3.2 Using Gray codes for sharing

Following Sutskever and Hinton (2008), all that is required to build a universal approximator is the ability to transfer probability mass from a_k to a_{k+1} while leaving the probability of the other vectors unchanged, and this for all values of k. The goal of the following sections and associated theorems, and one of the contributions of this paper, is to show how, given appropriate sequences $(a_k)_{1 \leq k \leq 2^n}$, we can implement this "sharing" in an efficient way, that is using a one-layer sigmoid belief network of size n instead of a three-layer network of size $n + 1$.

The sequences we will use are so-called Gray codes, which are sequences $(a_k)_{1 \leq k \leq 2^n}$ such that

- $\cup_k \{a_k\} = \{0, 1\}^n$
- $\forall k$ s.t. $2 \leq k \leq 2^n$, $\|a_k - a_{k-1}\|_H = 1$ where $\| \cdot \|_H$ is the Hamming distance.

where $\cup$ indicates logical or. There are many such codes Gray (1953).

3.3 Single Sequence

Here we go through the 2^n configurations in a particular order, following a Gray code, i.e., such that only one bit is changed at a time.

Let us consider two consecutive layers $\mathbf{h}$ and $\mathbf{v}$ (that is there is some r such that $\mathbf{h} = \mathbf{h}^{r+1}$ and $\mathbf{v} = \mathbf{h}^r$) of size n with W_{ij} the weight linking unit v_i to unit h_j, b_i the bias of unit v_i and w a positive scalar.

Let us first quickly remind that, for every positive scalar ε ($0 < \varepsilon < 1$), there is a weight vector $W_{i,:}$ and a real b_i such that $P(v_i|\mathbf{h}) = (1-\varepsilon)\mathbf{1}_{v_i=h_i}$ (that is, the i-th bit of $\mathbf{h}$ is copied to $\mathbf{v}$ with probability $1-\varepsilon$). Indeed, setting:

- $W_{ii} = 2w$
- $W_{ij} = 0$ for $i \neq j$
- $b_i = -w$

yields a total input to unit v_i of:

$$I(v_i, \mathbf{h}) = 2wh_i - w \qquad . \tag{3}$$

Therefore, if $w = \text{sigm}^{-1}(1-\varepsilon) = \log \frac{1-\varepsilon}{\varepsilon}$, we have $P(v_i = h_i|\mathbf{h}) = (1-\varepsilon)$.

Having proven this, we move on to the next - less obvious - theorem, in order to control the transfer of probability mass for one input pattern, with a single added layer.

Theorem 1. *Let $\mathbf{a}_t$ be an arbitrary binary vector in $\{0,1\}^n$ with its last bit equal to 0 and p a scalar. For every positive scalar ε ($0 < \varepsilon < 1$), there is a weight vector $W_{n,:}$ and a real b_n such that:*

- *if the binary vector $\mathbf{h}$ is not equal to $\mathbf{a}_t$, the last bit remains unchanged with probability greater than or equal to $1-\varepsilon$, that is $P(v_n = h_n|\mathbf{h} \neq \mathbf{a}_t) > (1-\varepsilon)$.*
- *if the binary vector $\mathbf{h}$ is equal to $\mathbf{a}_t$, its last bit is switched from 0 to 1 with probability $\text{sigm}(p)$.*

Proof. For the sake of simplicity and without loss of generality, we will assume that the first k bits of $\mathbf{a}_t$ are equal to 1 and that the remaining $n-k$ are equal to 0, with $k < n$.

We will now define the weights and biases as follows:

- $W_{nj} = w, 1 \leq j \leq k$
- $W_{nj} = -w, k+1 \leq j \leq n-1$
- $W_{nn} = nw$
- $b_n = -kw + p$

The total input to v_n is

$$I(v_n, \mathbf{h}) = w \left(\sum_{j=1}^{k} h_j - \sum_{j=k+1}^{n-1} h_j + nh_n - k \right) + p \tag{4}$$

If $\mathbf{h} = \mathbf{a}_t$, then

$$I(v_n, \mathbf{a}_t) = p \tag{5}$$

Otherwise, there are two possibilities:

1. $h_n = 0$, in which case $I(v_n, h) \leq -w + p$
2. $h_n = 1$, in which case $I(v_n, h) \geq w + p$

Again, if w is greater than $\mathrm{sigm}^{-1}(1-\varepsilon) + |p| = \log \frac{1-\varepsilon}{\varepsilon} + |p|$ and $\mathbf{h}$ is different from $\mathbf{a}_t$, then we have $P(v_n = h_n | \mathbf{h} \neq \mathbf{a}_t) > (1-\varepsilon)$.

Summing up, the transformation performed by these parameters is:

- if the vector $\mathbf{h}$ is different from $\mathbf{a}_t$, leave the last bit unchanged with probability greater than $1-\varepsilon$,
- if the vector $\mathbf{h}$ is equal to $\mathbf{a}_t$, flip its last bit from 0 to 1 with probability $\mathrm{sigm}(p)$.

This concludes the proof. □

It is easy to change the parameters so that the flip would be from 1 to 0. We would simply need to set:

- $W_{nj} = -w, 1 \leq j \leq k$
- $W_{nj} = w, k+1 \leq j \leq n-1$
- $W_{nn} = nw$
- $b_n = (k-n)w + p$

This could of course be done with any bit and not just the last one.

We have proven that, for any layer, it it possible to keep all the bits but one unchanged (with a probability arbitrarily close to 1) and to change the remaining one (with some probability) only when $\mathbf{h}$ matches a certain vector. Consequently, if we find a sequence of $(a_i)_{1 \leq i \leq 2^n}$ such that the difference between a_i and a_{i+1} is only one bit, following the proof of Sutskever and Hinton (2008), we will have built a universal approximator. A Gray code is such a sequence.

3.4 Multiple Sequences

The previous method still requires $2^n + 1$ layers of size n, which brings the total number of parameters to $n^2 \cdot (2^n + 1)$, approximately n^2 times more than the number of degrees of freedom of the distribution and n times more than the number of parameters required to model the distribution with a single RBM (see Le Roux and Bengio (2008) for a proof).

The reader may have guessed where this factor n can be gained: instead of changing only one bit per layer (of size n), we should be able to change many of them, on the order of n. It would therefore be useful to build layers able to move from the k-th vector to the $k+1$-th vector of n different Gray codes rather than just one. If we manage to do so, at every layer, n new vectors will have the correct probability mass, making it only necessary to have $\frac{2^n}{n}$ layers.

To achieve that goal, we will need to build a slightly more complicated layer. Let us again consider a sequence of two consecutive layers $\mathbf{h}$ and $\mathbf{v}$ of size n with W_{ij} the weight linking unit v_i to unit h_j, b_i the bias of unit v_i and w a positive scalar.

Theorem 1 showed that a sequence of two consecutive layers could keep all the vectors but one unchanged (with probability arbitrarily close to 1) while flipping one bit of the last possible vector with some arbitrary probability. We will now show that, what theorem 1 achieved with one vector, can be achieved with two, provided the Hamming distance between these two vectors is exactly one.

Theorem 2. *Let $\mathbf{a}_t$ be an arbitrary binary vector in $\{0,1\}^n$, with last bit equal to 0 and $\mathbf{c}_t$ the vector obtained when switching the first bit of $\mathbf{a}_t$. Let p_0 and p_1 be two scalars and ε a positive scalar ($0 < \varepsilon < 1$). Then there is a weight vector $W_{n,:}$ and a scalar b_n*

such that:

- *If the vector* $\mathbf{h}$ *is not equal to* $\mathbf{a}_t$ *nor to* $\mathbf{c}_t$*, the last bit remains unchanged with probability greater than* $1-\varepsilon$*, that is* $P(v_n = h_n|\mathbf{h}) \geq (1-\varepsilon)$.
- *If the vector* $\mathbf{h}$ *is equal to* $\mathbf{a}_t$*, its last bit is flipped from 0 to 1 with probability* $\mathrm{sigm}(p_0)$.
- *If the vector* $\mathbf{h}$ *is* $\mathbf{c}_t$*, its last bit is flipped from 0 to 1 with probability* $\mathrm{sigm}(p_1)$.

Proof. Again, for the sake of simplicity and without loss of generality, we will assume that the first k bits of $\mathbf{a}_t$ are equal to 1 and that the remaining $n-k$ are equal to 0, with $k < n$.

We will now define the weights and biases as follows:

- $W_{n1} = p_0 - p_1$
- $W_{nj} = w$, $2 \leq j \leq k$
- $W_{nj} = -w$, $k+1 \leq j \leq n-1$
- $W_{nn} = nw$
- $b_n = -(k-1)w + p_1$

The total input to v_n is

$$I(v_n, h) = w\left(\sum_{j=2}^{k} h_j - \sum_{j=k+1}^{n-1} h_j + nh_n - (k-1)\right) + p_1 + (p_0 - p_1)h_1 \tag{6}$$

If $\mathbf{h}$ is equal to $\mathbf{a}_t$, then

$$I(v_n, \mathbf{a}_t) = p_0 \text{ since } h_1 = 1 \text{ for } \mathbf{a}_t. \tag{7}$$

If $\mathbf{h}$ is equal to $\mathbf{c}_t$, then

$$I(v_n, \mathbf{c}_t) = p_1 \text{ since } h_1 = 0 \text{ for } \mathbf{c}_t. \tag{8}$$

Otherwise, there are two possibilities:

1. $h_n = 0$, in which case $I(v_n, h) \leq -w + \max(p_0, p_1)$,

2. $h_n = 1$, in which case $I(v_n, h) \geq 2w + \min(p_0, p_1)$.

If w is equal to $\max(0, \log \frac{1-\varepsilon}{\varepsilon} + \max(|p_0|, |p_1|))$, we have

$$\begin{aligned}
P(v_n = 1|\mathbf{h}, h_n = 0) &\leq \text{sigm}(-w + \max(p_0, p_1)) \\
&\leq \text{sigm}(-\log \frac{1-\varepsilon}{\varepsilon}) \\
&= \varepsilon \\
P(v_n = 1|\mathbf{h}, h_n = 1) &\geq \text{sigm}(2w + \min(p_0, p_1)) \\
&\geq \text{sigm}(2\max(0, \log \frac{1-\varepsilon}{\varepsilon} + \max(|p_0|, |p_1|)) + \min(p_0, p_1)) \\
&\geq \text{sigm}(\log \frac{1-\varepsilon}{\varepsilon}) \\
&= 1 - \varepsilon
\end{aligned}$$

Therefore, if $\mathbf{h}$ is different from $\mathbf{a}_t$ and from $\mathbf{c}_t$, then $P(v_n = h_n|\mathbf{h}) \geq 1 - \varepsilon$. This concludes the proof. □

Figure 2 shows such a layer.

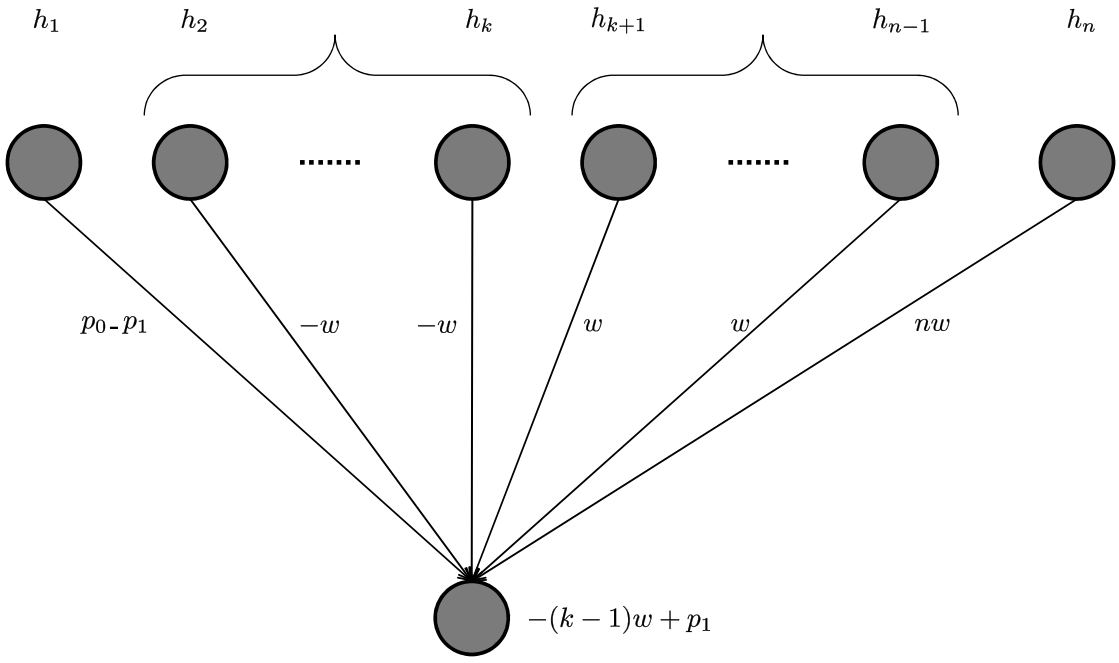

Figure 2: Representation of the layers used in th. 2.

We will now consider n Gray code sequences in parallel, allowing ourselves to transfer probability mass to n new vectors at each layer.

We will focus on sequences of n bits where n is a power of 2, i.e. $n = 2^t$.

Theorem 3. *Let $n = 2^t$. There exist n sequences of vectors of n bits S_i, $0 \leq i \leq n-1$ composed of vectors $S_{i,k}$, $1 \leq k \leq \frac{2^n}{n}$ satisfying the following conditions:*

1. *$\{S_0, \ldots, S_{n-1}\}$ is a partition of the set of all vectors of n bits.*
2. *For every i in $\{0, \ldots, n-1\}$ and every k in $\{1, \ldots, \frac{2^n}{n}-1\}$, the Hamming distance between $S_{i,k}$ and $S_{i,k+1}$ is 1.*
3. *For every $\{i, j\}$ in $\{0, \ldots, n-1\}^2$ such that $i \neq j$, and for every k in $\{1, \ldots, \frac{2^n}{n} - 1\}$, the bit switched between $S_{i,k}$ and $S_{i,k+1}$ and the bit switched between $S_{j,k}$ and $S_{j,k+1}$ are different, unless the Hamming distance between $S_{i,k}$ and $S_{j,k}$ is 1.*

Proof. We will prove this theorem by construction.
Let G_{n-t} be a Gray code over $n-t$ bits and G^i_{n-t} the same Gray code where every vector has been shifted by i bits to the right (the i rightmost bits being at the beginning of the vector). For instance,

$$G_2 = G_2^0 = \begin{matrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{matrix} \qquad G_2^1 = \begin{matrix} 0 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{matrix}$$

The first t bits of every vector in the sequence S_i will be the binary representation of i over t bits. For $0 \leq i < \frac{n}{2}$, the last $n-t$ bits of S_i will be G^i_{n-t}. For $\frac{n}{2} \leq i < n$, the last $n-t$ bits of S_i will be $G^{i-\frac{n}{2}}_{n-t}$. We emphasize that one shall not confuse the index of the sequence (which runs from 0 to $n-1$) with the shift of the Gray code (which runs from 0 to $\frac{n}{2}-1$). Therefore, no sequence is shifted by more than $\frac{n}{2}-1$ bits to the right. Here are the four sequences for $t = 2$:

$$S_0 = \begin{matrix} (0 & 0 & 0 & 0) \\ (0 & 0 & 0 & 1) \\ (0 & 0 & 1 & 1) \\ (0 & 0 & 1 & 0) \end{matrix} \quad S_1 = \begin{matrix} (0 & 1 & 0 & 0) \\ (0 & 1 & 1 & 0) \\ (0 & 1 & 1 & 1) \\ (0 & 1 & 0 & 1) \end{matrix} \quad S_2 = \begin{matrix} (1 & 0 & 0 & 0) \\ (1 & 0 & 0 & 1) \\ (1 & 0 & 1 & 1) \\ (1 & 0 & 1 & 0) \end{matrix} \quad S_3 = \begin{matrix} (1 & 1 & 0 & 0) \\ (1 & 1 & 1 & 0) \\ (1 & 1 & 1 & 1) \\ (1 & 1 & 0 & 1) \end{matrix}$$

One can see that condition 1 is satisfied. Indeed, let $\mathbf{x}$ be a vector over n bits. Let i be the value represented by its first t bits. Since $0 \leq i \leq n-1$ (because $n = 2^t$), the first t bits of $\mathbf{x}$ match the first t bits of every vector in S_i. Then, no matter what its

remaining $n-t$ bits are, they will appear exactly once in the code G^i_{n-t} since such a Gray code lists all the vectors of $n-t$ bits. Therefore, $\mathbf{x}$ will appear exactly once in S_i and will not appear in the other sequences.

Condition 2 is trivially satisfied by construction since, within each sequence, the first t bits do not change and the last $n-t$ bits form a Gray code.

Since a Gray code only changes one bit at a time, for every k in $1, \ldots, 2^{n-t}-1$, the bit change between the k-th vector and the $k+1$-th vector of G^i_{n-t} and between the k-th vector and the $k+1$-th vector of G^j_{n-t} is different, unless $i-j \equiv 0(\ \mathbf{mod}\ n-t)$. Since $0 \leq i < \frac{n}{2}$ and $\frac{n}{2} \leq n-t$ for $t \geq 1$, we only have $i-j \equiv 0(\ \mathbf{mod}\ n-t)$ for pairs of sequences $\{S_i, S_{i+\frac{n}{2}}\}$, $0 \leq i < \frac{n}{2}$. Such sequences share the same Gray code on the last $n-t$ bits and their first t bits only differ in one position (the first one). Therefore, condition 3 is also satisfied. □

Before proving the last theorem, we will introduce the following lemma, which gives us the correct probabilities to assign to each vector at each layer:

Lemma 1. *Let p^* be an arbitrary distribution over vectors of n bits, where n is again a power of two. A DBN with $\frac{2^n}{n}+1$ layers such that:*

1. *for each i, $0 \leq i \leq n-1$, the top RBM between layers $\mathbf{h}^{\frac{2^n}{n}}$ and $\mathbf{h}^{\frac{2^n}{n}-1}$ assigns probability $\sum_k p^*(S_{i,k})$ to $S_{i,1}$ where the $S_{i,k}$ are the same as in th. 3.*

2. *for each i, $0 \leq i \leq n-1$ and each k, $1 \leq k \leq \frac{2^n}{n}-1$, we have*

$$P\left(\mathbf{h}^{\frac{2^n}{n}-(k+1)} = S_{i,k+1} \middle| \mathbf{h}^{\frac{2^n}{n}-k} = S_{i,k}\right) = \frac{\sum_{t=k+1}^{\frac{2^n}{n}} p^*(S_{i,t})}{\sum_{t=k}^{\frac{2^n}{n}} p^*(S_{i,t})} \tag{9}$$

$$P\left(\mathbf{h}^{\frac{2^n}{n}-(k+1)} = S_{i,k} \middle| \mathbf{h}^{\frac{2^n}{n}-k} = S_{i,k}\right) = \frac{p^*(S_{i,k})}{\sum_{t=k}^{\frac{2^n}{n}} p^*(S_{i,t})} \tag{10}$$

3. *for each k, $1 \leq k \leq \frac{2^n}{n}-1$, we have*

$$P\left(\mathbf{h}^{\frac{2^n}{n}-(k+1)} = \mathbf{a} \middle| \mathbf{h}^{\frac{2^n}{n}-k} = \mathbf{a}\right) = 1 \text{ if } \mathbf{a} \notin \cup_i S_{i,k} \tag{11}$$

has p^* as its marginal distribution over $\mathbf{h}^0$.

Proof. Let $\mathbf{x}$ be an arbitrary vector over n bits. According to theorem 3, there is a pair (i,k) such that $\mathbf{x} = S_{i,k}$. This DBN is such that, for all i and all k, if $\mathbf{h}^{\frac{2^n}{n}-k} = S_{i,k}$,

then either $\mathbf{h}^{\frac{2^n}{n}-k-1} = S_{i,k}$ or $\mathbf{h}^{\frac{2^n}{n}-k-1} = S_{i,k+1}$. Therefore, to have $\mathbf{h}^0 = S_{i,k}$, all the hidden layers must contain a vector belonging to the i-th sequence. In fact, there is only one sequence which can lead to $\mathbf{h}^0 = S_{i,k}$. It is:

- $\mathbf{h}^{\frac{2^n}{n}-t} = S_{i,t}$ for $0 \leq t \leq k$
- $\mathbf{h}^{\frac{2^n}{n}-t} = S_{i,k}$ for $k \leq t \leq \frac{2^n}{n}$

The marginal probability of $\mathbf{h}^0 = S_{i,k}$ is therefore the probability of such a sequence, which is equal to

$$
\begin{aligned}
p(\mathbf{h}^0 = S_{i,k}) &= P(\mathbf{h}^{\frac{2^n}{n}-1} = S_{i,1}) \\
&\quad \prod_{t=1}^{k-1} P\left(\mathbf{h}^{\frac{2^n}{n}-(t+1)} = S_{i,t+1} \,\middle|\, \mathbf{h}^{\frac{2^n}{n}-t} = S_{i,t}\right) \\
&\quad P\left(\mathbf{h}^{\frac{2^n}{n}-(k+1)} = S_{i,k} \,\middle|\, \mathbf{h}^{\frac{2^n}{n}-k} = S_{i,k}\right) \\
&\quad \prod_{t=k+1}^{\frac{2^n}{n}-1} P\left(\mathbf{h}^{\frac{2^n}{n}-(t+1)} = S_{i,k} \,\middle|\, \mathbf{h}^{\frac{2^n}{n}-t} = S_{i,k}\right) \qquad (12) \\
&= \left(\sum_{u=1}^{\frac{2^n}{n}} p^*(S_{i,u})\right) \prod_{t=1}^{k-1} \frac{\sum_{u=t+1}^{\frac{2^n}{n}} p^*(S_{i,u})}{\sum_{u=t}^{\frac{2^n}{n}} p^*(S_{i,u})} \\
&\quad \frac{p^*(S_{i,k})}{\sum_{u=k}^{\frac{2^n}{n}} p^*(S_{i,u})} \cdot 1^{\frac{2^n}{n}-1-k} \qquad (13) \\
&= p^*\left(S_{i,k}\right) \qquad (14)
\end{aligned}
$$

The last result stems from the cancelation of consecutive terms in the product. This concludes the proof. □

This brings us to the last theorem:

Theorem 4. *If $n = 2^t$, a DBN composed of $\frac{2^n}{n} + 1$ layers of size n is a universal approximator of distributions over vectors of size n.*

Proof. Using lemma 1, we now show that it is possible to construct such a DBN.

First, Le Roux and Bengio (2008) showed that an RBM with n hidden units can model any distribution which assigns a non-zero probability to at most n vectors. Property 1 of lemma 1 can therefore be achieved.

All the subsequent layers are as follows.

- At each layer, the first t bits of $\mathbf{h}^{k+1}$ are copied to the first t bits of $\mathbf{h}^k$ with probability arbitrarily close to 1. This is possible as proven in section 3.3.

- At each layer, $n/2$ of the remaining $n-t$ bits are potentially changed to move from one vector in a Gray code sequence to the next with the correct probability (as defined in lemma 1). Each of these $n/2$ bits will only change if the vector on $\mathbf{h}^{k+1}$ matches one of two possibilities (cf th. 3), which is possible (cf th. 2).

- The remaining $n/2-t$ bits are copied from $\mathbf{h}^{k+1}$ to $\mathbf{h}^k$ with probability arbitrarily close to 1.

Such layers are arbitrarily close to fulfilling the requirements of the second property of lemma 1. This concludes the proof. □

4 Universal discriminative model

In this section we exploit the proof technique developed above in order to prove universal approximation properties for deep but narrow feedforward neural network binary classifiers with binary inputs, that is **every function from $H_n = \{0,1\}^n$ to $\{0,1\}$ can be modeled by a feedforward neural network composed of $2^{n-1}+1$ layers of size n.**

One must note that the universal approximation property of deep and narrow feedforward neural networks has already been proven in Rojas (2003), which shows that one may solve any two-class classification problem using nested convex polytopes, each of these being modeled by a stack of perceptrons. The final result is that one can model any function from $\mathbb{R}^n$ to $\{0,1\}$ using a feedforward neural network, provided that:

- each layer receives one bit of information from the layer below

- each layer is connected to the input.

If we were to build an equivalent network where each layer is only connected to the layer below, then we would need $n+1$ hidden units (n for the input and one for the extra bit of information provided by the layer below in his architecture) per layer. This section tries to prove the same property with hidden layers of size n.

As in section 3.3, we will first consider a sequence of two consecutive layers $\mathbf{h}$ and $\mathbf{v}$ of size n with W_{ij} the weight linking unit v_i to unit h_j and b_i the bias of unit v_i. The model is a sigmoid belief network directed from $\mathbf{h}$ to $\mathbf{v}$.

Theorem 5 (Arbitrary projection)**.** *Let $W_{0,:}$ be a vector of $\mathbb{R}^n$, b_0 a scalar, ε a positive scalar ($0 < \varepsilon < 1$) and*

$$S = \left\{ \mathbf{h} \in H_n \left| W_{0,:}^T \mathbf{h} + b_0 > 0 \right. \right\} \tag{15}$$

where H_n is the binary hypercube in dimension n.

Then, for all i, $1 \leq i \leq n$, there exists a weight vector $W_{i,:}$ and a scalar b_i such that

$$\begin{cases} \textit{if } \mathbf{h} \in S, P(v_i = 1|\mathbf{h}) > 1 - \varepsilon \\ \textit{if } \mathbf{h} \notin S, P(v_i = h_i|\mathbf{h}) > 1 - \varepsilon \end{cases}$$

Proof. Let us define:

$$\begin{aligned} t_1 &= \min_{\mathbf{h} \in S} W_{0,:}^T \mathbf{h} + b_0 \\ t_2 &= \min_{\mathbf{h} \in H_n} W_{0,:}^T \mathbf{h} + b_0 \end{aligned}$$

Since S is a finite set, t_1 is strictly positive. Since S is included in H_n, $\dfrac{t_2}{t_1} \leq 1$. Let w be a positive scalar. Defining the weights and bias as follows:

- $b_i = w \left(\frac{b_0}{t_1} - \frac{1}{2} \right)$
- $W_{ii} = w \left(\frac{W_{0i}}{t_1} + 1 - \frac{t_2}{t_1} \right)$
- $W_{ij} = w \frac{W_{0j}}{t_1}, j \neq i$

we have

$$\begin{aligned} P(v_i = 1|\mathbf{h}) &= \text{sigm} \left(\sum_j W_{ij} h_j + b_i \right) \\ &= \text{sigm} \left(w \left[\sum_{j \neq i} \frac{W_{0j}}{t_1} h_j + \left(\frac{b_0}{t_1} - \frac{1}{2} \right) + \left(\frac{W_{0i}}{t_1} + 1 - \frac{t_2}{t_1} \right) h_i \right] \right) \\ P(v_i = 1|h) &= \text{sigm} \left(w \left[\frac{W_{0,:}^T h + b_0}{t_1} - \frac{1}{2} + \left(1 - \frac{t_2}{t_1} \right) h_i \right] \right) \end{aligned}$$

Therefore,

$$\begin{cases} \text{if } \mathbf{h} \in S, \quad P(v_i = 1|\mathbf{h}) \geq \text{sigm}\left(w\left[\frac{1}{2} + \left(1 - \frac{t_2}{t_1}\right) h_i\right]\right) \\ \text{if } \mathbf{h} \notin S, P(v_i = 1|\mathbf{h}) \geq \text{sigm}\left(w\left[\frac{t_2}{t_1} - \frac{1}{2} + \left(1 - \frac{t_2}{t_1}\right) h_i\right]\right) \\ \text{if } \mathbf{h} \notin S, \ P(v_i = 1|\mathbf{h}) \leq \text{sigm}\left(w\left[-\frac{1}{2} + \left(1 - \frac{t_2}{t_1}\right) h_i\right]\right) \end{cases}$$

The last equation stems from the fact, that if $\mathbf{h}$ is not in S, then $W_{0,:}^T \mathbf{h} + b_0 \leq 0$.

Since $\frac{t_2}{t_1} \leq 1$, $\frac{1}{2} + \left(1 - \frac{t_2}{t_1}\right) h_i$ is always strictly positive no matter what the value of h_i is. Thus, if $\mathbf{h}$ is in S, the argument of the sigmoid is always strictly positive.

If $\mathbf{h}$ is not in S and $h_i = 0$, then $P(v_i = 1|h) \leq \text{sigm}\left(-\frac{w}{2}\right)$.

If $\mathbf{h}$ is not in S and $h_i = 1$, then $P(v_i = 1|h) \geq \text{sigm}\left(\frac{w}{2}\right)$.

When w tends to $+\infty$, these probabilities tend to 1. Therefore, for all ε such that $0 < \varepsilon < 1$, there exists a scalar C such that, if $w > C$, these probabilities are larger than $1 - \varepsilon$. □

It is trivial to adapt theorem 5 so that

$$\begin{cases} \text{if } h \in S, P(v_i = 0|h) = 1 \\ \text{if } h \notin S, P(v_i = h_i|h) = 1 \end{cases}$$

Therefore, using this strategy for $1 \leq i \leq n$, we can apply the following transformation at every layer:

- define a vector $W_{0,:}$ and a bias b_0
- define $S = \left\{h \in H_n \left| W_{0,:}^T h + b_0 > 0 \right.\right\}$
- choose an h_0 in H_n
- for every $\mathbf{h}$ in S, map $\mathbf{h}$ to h_0
- for every $\mathbf{h}$ not in S, map $\mathbf{h}$ to itself

In the following theorem, and until the last stage, we shall use sets S which contain only one vector $\mathbf{h}$.

This allows us to prove the following theorem:

Theorem 6 (Universal discriminators)**.** *A neural network with $2^{n-1}+1$ layers of n units with the sigmoid as transfer function can model any non-constant function f from H_n to $\{0, 1\}$ arbitrarily well.*

Proof. Let N_0 be the number of vectors $\mathbf{h}$ such that $f(\mathbf{h}) = 0$ and N_1 be the number of vectors $\mathbf{h}$ such that $f(\mathbf{h}) = 1$. We therefore have:

- $N_0 + N_1 = 2^n$
- $\min(N_0, N_1) \leq 2^{n-1}$

Let us assume that $N_0 \leq N_1$ (and, subsequently, $N_0 \leq 2^{n-1}$). Let h_0 be a vector to be mapped to 0. At every layer, we will pick an arbitrary binary vector $\mathbf{h}$ such that $h \neq h_0$ and $f(h) = 0$ and map it to h_0, leaving the other vectors unchanged. This is possible using theorem 5. Once all the vectors to be mapped to 0 have been mapped to h_0 (which requires at most 2^{n-1} layers, including the input layer), the hyperplane separating h_0 from all the other vectors of H_n performs the correct mapping. □

5 Conclusion

Despite the surge in interest for deep networks in recent years, little is known about their theoretical power. We have introduced a proof technique based on Gray codes that allows to improve on a previous theorem Sutskever and Hinton (2008) regarding the representational power of deep but narrow sigmoidal belief networks (such as DBNs Hinton *et al.* (2006)). Instead of 3×2^n layers of size n, the bound presented here involves $\frac{2^n}{n} + 1$ layers of size n (i.e. $n^2 + n2^n + 2^n + n$ parameters). We do not know if this is the lowest achievable number of layers. Noticing that only half of the units at each layer may change, we believe this bound could be improved by a factor of 2 or less. One important thing to notice is that this is, perhaps unsurprisingly, of the same order of magnitude as the maximum number of parameters required in an RBM to model any distribution. This brings other, much more complex and yet more interesting questions: for a given number of parameters, which architecture can best represent distributions of interest? Is the representational power of DBNs more concentrated around real-world distributions when one only has access to a limited number of parameters, i.e. a limited number of training examples?

Finally, exploiting the same proof technique, we also showed that deep but narrow deterministic networks (with no more than $2^{n-1} + 1$ layers of size n) can represent any binary classifier on n-dimensional binary vectors.

References

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, **2, Issue 1**.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press.

Freund, Y. and Haussler, D. (1991). Unsupervised learning of distributions of binary vectors using 2-layer networks. In *NIPS*, pages 912–919.

Gray, F. (1953). Pulse code communication. U.S. Patent 2,632,058.

Hastad, J. and Goldmann, M. (1991). On the power of small-depth threshold circuits. *Computational Complexity*, **1**, 113–129.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, **14**, 1771–1800.

Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, **18**, 1527–1554.

Le Roux, N. and Bengio, Y. (2008). Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, **20**(6), 1631–1649.

Neal, R. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, **56**, 71–113.

Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press.

Rojas, R. (2003). Networks of width one are universal classifiers. In *International Joint Conference on Neural Networks*, volume 4, pages 3124–3127.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge.

Sutskever, I. and Hinton, G. E. (2008). Deep, narrow sigmoid belief networks are universal approximators. *Neural Computation*, **20**(11), 2629–2636.

Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the International Conference on Machine Learning*, volume 25.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'2008)*.