Department of Computer Science
University of Toronto
http://learning.cs.toronto.edu

6 King's College Rd, Toronto
M5S 3G4, Canada
fax: +1 416 978 1455

November 21, 2007

UTML TR 2007–002

Deep Narrow Sigmoid Belief Networks are Universal Approximators

Ilya Sutskever & Geoffrey E. Hinton
Department of Computer Science, University of Toronto

Abstract

In this paper we show that exponentially deep belief networks [3, 7, 4] can approximate any distribution over binary vectors to arbitrary accuracy, even when the width of each layer is limited to the dimensionality of the data. This resolves an open the problem in [6]. We further show that such networks can be greedily learned in an easy yet impractical way.

Deep Narrow Sigmoid Belief Networks are Universal Approximators

Ilya Sutskever & Geoffrey E. Hinton
Department of Computer Science, University of Toronto

1 Introduction

In [3] a fast greedy algorithm for learning deep belief networks was introduced. The algorithm uses a Restricted Boltzmann Machine (RBM) (see e.g., [2] and references therein) to learn a model of the input data. The learned RBM's hidden variables are used to transform the original data distribution to a new, transformed distribution, which is typically easier to model. The transformed distribution is learned by a new RBM, which, after completing its learning, further transforms the already transformed distribution. A recursive repetition of this process is the essence of the greedy learning algorithm for deep belief networks. It is shown in [3] that this algorithm increases a lower bound on the log likelihood of a deep belief network whose prior distribution over the topmost layer is defined by an RBM, justifying the algorithm from a statistical point of view. In practice, the greedy algorithm often learns deep belief networks that appear to fit the training data well.

In discriminative applications the greedy algorithm is used to initialize the network's parameters which are later fine-tuned using backpropagation [9] on a discriminative task. The greedy algorithm has been applied successfully to several real-world problems [4, 10], including state of the art digit recognition and document retrieval. The greedy algorithm has also been used to learn kernels that significantly improve the performance of Gaussian Processes on difficult tasks (compared to standard kernels) [11]. The potential of deep belief networks together with the limitations of kernel methods [1] have sparked an increased interest in understanding the capabilities and properties of deep belief networks.

An obvious question posed in [6] is whether deep belief networks can approximate any distribution to arbitrary precision even when their width is limited.

Existing approximation results are unable to answer this question because they are only applicable to neural networks with a single layer of exponential size (see e.g., [5, 6]) or for an exponentially deep *feedforward* network computing an input-output mapping [8].

In this work we positively resolve this question. We show that for any distribution over n-dimensional binary vectors there is a deep belief network with maximal layer width of size $n+1$ and depth $3 \cdot (2^n - 1) + 1$ that approximates the distribution to any desired precision. We also show that adding hidden layers always increases the representational power of the deep belief network unless the network is already exponentially deep. In addition, we introduce a simple greedy learning algorithm for learning a network approximating any distribution.

2 Deep Belief Networks

Before we describe the constructions, we define deep belief networks (henceforth, we will use the terms sigmoid belief networks and deep belief networks interchangeably). Sigmoid belief networks are described in full generality in [7]. In this paper we restrict ourselves to layered sigmoid belief networks, in which the observed units (the outputs) are at the lowest layer.

Let $g(x) = (1+\exp(-x))^{-1}$ be the logistic function, $logit(y)$ be g's inverse function, so $logit(g(x)) = x$ for all x, and $V_0, \ldots, V_N$ be a sequence of random variables each of which is a binary vector, such that V_0 is the visible layer where the outputs of the generative model are observed and $V_1, \ldots, V_N$ are the hidden layers. Each coordinate of V_k is also called a unit.

We consider probability distributions of the form

$$P(V_0 = v_i, \ldots, V_N = v_N) = \prod_{i=0}^{N-1} P_i(V_i = v_i | V_{i+1} = v_{i+1}) P_N(V_N = v_N) \tag{1}$$

so that the distribution defined by P on V_0 is just $P(V_0)$, which is the result of marginalizing $V_1, \ldots, V_N$.

This distribution is that of a sigmoid belief network if for all i, $P_i(V_i|V_{i+1})$ is a factorial distribution with $P_i((V_i)_j = 1|V_{i+1}) = g((W_i \cdot v_{i+1} + b_i)_j)$, where W_i and b_i are the parameters of P_i: W_i is a matrix of connection weights between layers V_i and V_{i+1} and b_i is a vector of biases for V_i. Equivalently,

$$P_i(V_i = v_i | V_{i+1} = v_{i+1}) = \prod_{j=1}^{d_i} g((2(v_i)_j - 1) \cdot (W_i \cdot v_{i+1} + b_i)_j) \tag{2}$$

and

$$P_N(V_N = v_N) = \prod_{j=1}^{d_N} g((2(v_N)_j - 1) \cdot (b_N)_j) \tag{3}$$

In these equations, d_k denotes the dimensionality of V_k. We call $P(V_0)$ the *visible distribution* of the sigmoid belief network.

We define the *total input* of a variable $(V_i)_j$ to be $(W_i \cdot v_{i+1} + b_i)_j$. Note that when the total input is very large and positive $(V_i)_j$ is extremely likely to take the value 1; if it is very large and negative it is extremely likely to take the value 0.

3 The Construction

3.1 The Basic Idea

Given an arbitrary distribution assigning non-zero probability to a subset of binary vectors $\{x_0, \ldots, x_M\}$, we might take a fraction of the probability mass from x_0, and give it to, say, x_{M+1}, that is not in the subset. If we are not restricted in the choice of x_{M+1} and the fraction of the mass taken from x_0 to x_{M+1}, then *any* distribution can be constructed by repeatedly applying this rule with different x_{M+1}'s and different fractions. We call such a transformation of a distribution *sharing.*

Suppose, for example, that we want to apply sharing steps to get a distribution over the four binary vectors $00, 01, 10, 11$ with probabilities $(.5, .2, .1, .2)$, i.e., $Pr(00) = .5, Pr(01) = .2, Pr(10) = .1, Pr(11) = .2$, where we let $x_0 = 00$. We start with the initial distribution $(1, 0, 0, 0)$, and execute the following sharing steps:

- Distribution: $(1, 0, 0, 0)$ (initial)
  - Operation: give 2/10 of the mass of $x_0 = 00$ to $x_1 = 01$
- Distribution: $(.8, .2, 0, 0)$
  - Operation: give 1/8 of the mass of $x_0 = 00$ to $x_2 = 10$
- Distribution: $(.7, .2, .1, 0)$
  - Operation: give 2/7 of the mass of $x_0 = 00$ to $x_3 = 11$

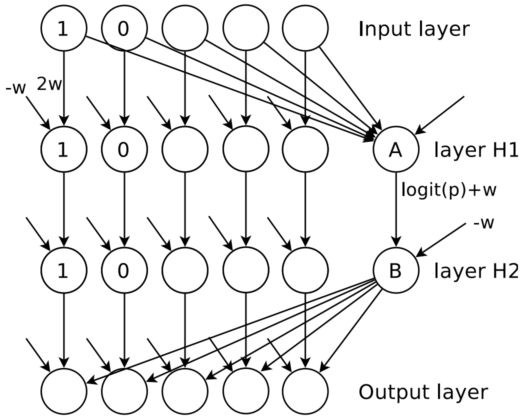

Figure 1: A Sigmoid Belief Network implementing sharing. If the input is not x_0 then $A = 0$ and the output is equal to the input. If the input is x_0 then $A = 1$, so $B = 1$ with probability p. If $B = 1$ then the output is equal to x_{M+1}. $B = 0$ with probability $1 - p$, in which case the output is equal to x_0.

- Distribution: $(.5, .2, .1, .2)$ (final)

Thus, to show that a sigmoid belief network can approximate any distribution, it is enough to show that a sigmoid belief network can implement sharing.

3.2 Implementing Sharing with a Sigmoid Belief Network

We now show how a sigmoid belief network can approximate an arbitrary sharing step to arbitrary accuracy using 3 layers. More specifically, we implement a distribution transformation that gives a fraction p of x_0's probability mass to x_{M+1} and leaves all the other probabilities unchanged, where x_0, x_{M+1}, and the fraction p are arbitrary. It is done it by implementing the stochastic mapping $Input \rightarrow Output$ (where $Input$ and $Output$ are n-dimensional binary vectors) such that if $Input \neq x_0$ then $Output = Input$, and if $Input = x_0$ then $Output = x_{M+1}$ with probability p and $Output = x_0$ with probability $1 - p$.

Consider figure 1. In the figure, there are four layers, $Input, H_1, H_2, Output$. Every pair of connected nodes acts as a flip-flop unit (except A and B), so that a varibale is equal to its parent with high probability, causing the output layer to be equal to the input layer whenever $B = 0$. This is done by setting the weights connecting the flip-flop units to $2w$ for some large w and and setting the bias to $-w$. Increasing w allows us to make the failure probability of each flip-flop arbitrarily small.

The variable A is equal to 1 if and only if the input layer is equal to x_0. It is implemented using a linear classifier that separates x_0 from the rest of the binary vectors with a positive margin. By multiplying the classifier's weights by a large a factor w, this margin can be made as large as desired, causing A to be equal to 1 with overwhelming probability if the input vector is equal to x_0, and 0 with overwhelming probability otherwise.

We let B decide whether probability mass should be given to x_{M+1}: if $B = 1$ then the output equals to x_{M+1}, but if $B = 0$ then the output equals the input. If $A = 1$ then the input layer is equal to x_0, so we set $B = 1$ with probability p, and when $A = 0$ then the input layer is not x_0, so we set $B = 0$. This is implemented by letting B have a large negative bias, $-w$, and setting the connection from A to B to the weight $w + logit(p)$ ($logit$ is the inverse of g). This way, if $A = 0$, B receives total input $-w$, which causes it to be 0 with very high probability, but if $A = 1$, B receives total input of size $logit(p)$, so it is equal to 1 with probability $g(logit(p)) = p$.

B is connected to the output layer with weights of absolute value of size $4w$, so that when $B = 1$, the output layer is set to x_{M+1} regardless the values of the flip-flop units in the layer above, but if $B = 0$ then the output layer is equal to the input layer.

This implements sharing: if the input is x_0, $A = 1$, so $B = 1$ with probability p which causes the output to be x_{M+1} with probability p; however, with probability $1 - p$ the output pattern stays equal to x_0. This is how x_0's probability is given to x_{M+1}. Any other pattern ($\neq x_0$) does not activate A and thus gets copied to the output layer, so the sharing implementation does not change the probabilities of every vector that is not x_0 or x_{M+1}.

The construction is completed by specifying $P_N(V_N)$, which assigns overwhelming probability to the zero vector[1].

Note that $2^n - 1$ sharing steps are sufficient to obtain any distribution over n-dimensional binary vectors, and the output layer of one sharing step implementation is the input layer of the next sharing step implementation, so there are $3(2^n - 1) + 1$ layers (the $+1$ term exists because of the distribution P_N). The approximation can be made arbitrarily accurate by making w large.

3.3 Adding Hidden Layers Increases Representational Power

Consider sigmoid belief networks with k layers of size $n+1$ (the visible layer V_0 is also of size $n+1$). Let D_k be the set of all distributions over $n+1$-dimensional binary vectors that can be approximated arbitrarily well by a sigmoid belief network of this size. For each distribution in D_k, we compute its marginal distribution over its first n dimensions and get a set of marginal distributions over n-dimensional binary vectors which we call D'_k.

It was known that adding hidden layers does not reduce the representational power of sigmoid belief networks (i.e., $D_k \subseteq D_{k+1}$) [3], but it was not known whether they increased it (i.e., $D_k \subset D_{k+1}$) [6]. We will show that unless $D'_k = ALL_n$, the set of all distributions over n-dimensional binary vectors, then $D_k \neq D_{k+1}$.

For the proof, suppose that $D_k = D_{k+1}$, namely, that for any sigmoid belief network with $k+1$ layers there is a sigmoid belief network with k layers with the same marginal distribution over the visible vectors V_0. From this it follows that $D_{k+2} = D_{k+1}$, since given a sigmoid belief network of depth $k+2$, we can replace the top $k+1$ hidden layers (i.e., $V_{k+2}, \ldots, V_1$) with k hidden layers (i.e., $V_{k+1}, \ldots, V_1$) such that the marginal distribution on V_1 is the same for both networks (because $D_k = D_{k+1}$ and all the layers are of size $n+1$). If we do not change the conditional probability of V_0 given V_1, we get the same marginal distribution on V_0 but with $k+1$ layers instead of $k+2$. Repeating this argument proves that $D_k = D_{3(2^n-1)+1}$, and we have demonstrated that $D'_{3(2^n-1)+1} = ALL_n$ in the previous section. So unless $D'_k = ALL_n$, $D_k \neq D_{k+1}$.

$D'_k = ALL_n$ is a strong condition that means that networks with $k-1$ layers of size $n+1$ and a visible layer of size n can approximate any distribution over n-dimensional binary vectors. If this condition is not met, then there is a deep belief network with $k+1$ layers each of size $n+1$ (none of size n) whose marginal distribution over V_0 cannot be approximated by a deep belief network with k layers each of size $n+1$.

This argument fails if in the definition of D_k, V_0 is an n-dimensional binary vector and $V_1, V_2, \ldots$ are $n+1$-dimensional, because it could no longer be argued that if $D_k = D_{k+1}$ then $D_{k+1} = D_{k+2}$, because the sigmoid belief network that is replaced has $n+1$ units in its visible vector and not n.

4 A Greedy Version of the Construction

The construction above is "top-down", while the greedy learning algorithm that motivated it is "bottom-up": an RBM learns the data distribution, transforms it, and lets another RBM learn and

[1] Repeated applications of sharing can actually transform any distribution into any other distribution, so this specification of P_N is not essential.

transform the transformed distribution, repeating this process as often as needed. In this section we show how a deep belief network approximating a distribution can be learned by a greedy, bottom-up algorithm that uses autoencoders with hidden layers instead of RBMs.

We define the *complexity* of a distribution to be the number of configurations to which the distribution assigns non-zero probability (i.e., the size of the support of the distribution).

Let V and H be random n-dimensional binary vectors. *Collapsing*, to be defined shortly, is a way to reduce the complexity of a distribution by 1. Assume that the data distribution on V assigns probabilities $p_0, \ldots, p_M$ to $x_0, \ldots, x_M$. To collapse this distribution, apply the deterministic function $H = f(V)$ to V, where $f(V) = V$ unless $V = x_M$, in which case $f(V) = x_0$. As a result, the distribution over H assigns non-zero probability to only $x_0, \ldots, x_{M-1}$ (but not x_M), x_0 has probability $p_0 + p_M$ under the collapsed distribution, and the probabilities of $x_1, \ldots, x_{M-1}$ are unchanged.

Collapsing is a special case of sharing where x_M gives all of its probability mass to x_0, and can be easily undone by a sharing step that takes the appropriate amount of probability mass from x_0 to x_M. Since the distribution on H has lower complexity than V, the pair of collapsing and sharing "gently models" the distribution on V.

Since the complexity of any distribution over n-dimensional binary vectors is bounded by 2^n, repeated (i.e., greedy) applications of collapsing will eventually reduce the complexity of the distribution to 1, in which case the distribution can be represented by a set of biases that simply put all the probability mass on this vector. Because all the collapsing steps can be undone by appropriate sharing steps, the sequential process of undoing all the collapsing steps, starting with the simplest possible distribution (according to our complexity measure), is in fact the generative process of a greedily trained sigmoid belief network whose visible distribution is equal to the original, high-complexity distribution.

5 Conclusions and Open Questions

We have positively resolved the approximation properties of deep and narrow sigmoid belief networks. The first natural question that arises is whether every distribution in which every vector has non-zero probability can be *exactly* represented as a deep belief network. The requirement of each vector to have non-zero probability is necessary, since a sigmoid belief network always assigns non-zero probabilities to all configurations. The second question is regarding the necessary depth of the network: can it be shown that a deep and narrow (with width $n+c$) network of, say, $\ll 2^n/n^2$ layers cannot approximate every distribution? We believe the answer to be yes because then the network has less parameters than the number of parameters of a distribution (which is $2^n - 1$). What if the number of layers is of order $2^n/n^2$ (note that a network with $2^n/n^2$ layers has about 2^n parameters)? Can any distribution be approximated in that case? Finally, is it necessary to use hidden layers of width $n + 1$, or do hidden layers of width n suffice?

References

[1] Y. Bengio and Y. Le Cun. Scaling learning algorithms towards AI. In Léon Bottou, Olivier Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press, 2007.

[2] G.E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[3] G.E. Hinton, S. Osindero, and Y.W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, 2006.

[4] G.E. Hinton and R.R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

[5] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[6] N. Le Roux and Y. Bengio. Representational power of restricted boltzmann machines and deep belief networks. Technical Report 1294, DIRO, U. Montreal, April 2007.

[7] R.M. Neal. Learning stochastic feedforward networks. Technical report, CRG-TR-90-7, Department of Computer Science, University of Toronto, 1990.

[8] R. Rojas. Networks of width one are universal classifiers. *Proceedings of the International Joint Conference on Neural Networks*, 4:3124–3127, 2003.

[9] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by back-propagating errors. *Nature*, 323(99):533–536, 1986.

[10] R.R. Salakhutdinov and G.E. Hinton. Semantic Hashing. *SIGIR workshou on graphical models*, 2007.

[11] R.R. Salakhutdinov and G.E. Hinton. Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes. *Advances in Neural Information Processing systems (NIPS-20)*, 2007.