A Lower Bound for Perceptrons and an Oracle Separation of the $\mathrm{PP}^{\mathrm{PH}}$ Hierarchy

Christer Berg* Staffan Ulfberg*

November 23, 2023

Abstract

We show that there are functions computable by linear size boolean circuits of depth k that require super-polynomial size perceptrons of depth $k-1$, for $k < \log n/(6\log\log n)$.

This result implies the existence of an oracle A such that $\Sigma_k^{p,A} \not\subseteq \mathrm{PP}^{\Sigma_{k-2}^{p,A}}$ and in particular this oracle separates the levels in the $\mathrm{PP}^{\mathrm{PH}}$ hierarchy.

Using the same ideas, we show a lower bound for another function, which makes it possible to strengthen the oracle separation to $\Delta_k^{p,A} \not\subseteq \mathrm{PP}^{\Sigma_{k-2}^{p,A}}$.

1 Introduction

For ordinary boolean circuits (consisting of AND, OR, and NOT gates) of constant depth Sipser [10] showed, in 1983, that there are functions computable by linear size circuits of depth k that require super-polynomial size circuits of depth $k-1$. In 1986, Håstad [6, 7] showed that the same functions actually require exponential size circuits of depth $k-1$, and this result implies the existence of an oracle that separates the levels in the polynomial time hierarchy. The correspondence between circuits and the polynomial time hierarchy was first noted by Furst, Saxe, and Sipser [3].

There is a similar correspondence between the complexity class $\mathrm{PP}^{\mathrm{PH}}$ and *perceptrons*. A perceptron is a circuit with a single threshold gate at the top, whose inputs are the outputs of boolean circuits with AND, OR, and NOT gates, called the perceptron's sub-circuits. A depth k perceptron has sub-circuits of depth $k-1$. Perceptrons can compute majority and are thus more powerful than constant depth circuits [3].

In 1991, Green [4] extended a result by Håstad [6] and proved a lower bound for the size of constant depth perceptrons that compute parity. He used this result to show that there is an oracle that separates $\oplus\mathrm{P}$ from $\mathrm{PP}^{\mathrm{PH}}$.

Green [5] also discussed the question of whether there is an oracle that separates the levels in the $\mathrm{PP}^{\mathrm{PH}}$ hierarchy. This would follow from a sufficiently strong lower bound for the size of depth $k-1$ perceptrons computing functions computable by perceptrons of depth k. He proposed that such a bound would be obtained from an induction argument similar to that of Håstad which was used to show the corresponding result for ordinary circuits. In the induction step, the Håstad switching lemma would be used, and the basis, Green claimed, would be to compare depth 3 and depth 4 perceptrons; he was able to make this separation in the monotone case.

*Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-100 44 Stockholm, Sweden. *E-mail:* {berg,staffanu}@nada.kth.se.

In this paper we are able to show that there are functions computable by linear size boolean circuits of depth k that require super-polynomial size perceptrons of depth $k-1$, for $k < \log n/(6\log\log n)$, and exponential size perceptrons for constant k.

The key to making the proof work is to use a separation between polynomial size depth 2 perceptrons with bounded fan-in and general polynomial size depth 2 perceptrons as the basis for the induction. This separation follows from the "One-in-a-box" theorem by Minsky and Papert [9]. To use this basis, we need a somewhat stronger statement of Håstad's switching lemma. The statement as formulated in this paper actually follows from looking at the proof by Håstad more carefully.

We show that our result on perceptrons implies the existence of an oracle that separates the levels in the $\mathrm{PP}^{\mathrm{PH}}$ hierarchy, and in fact that there exists an A such that $\Sigma_k^{p,A} \not\subseteq \mathrm{PP}^{\Sigma_{k-2}^{p,A}}$.

The fact that our basis, i.e., the "One-in-a-box" theorem by Minsky and Papert, implies the oracle separation $\mathrm{NP}^{\mathrm{NP}} \not\subseteq \mathrm{PP}$ was noted by Fu [2]. Beigel [1] has strengthened this separation to obtain that $\mathrm{P}^{\mathrm{NP}} \not\subseteq \mathrm{PP}$, and in the last section of this paper we use his result as a basis for a lower bound for perceptrons with bounded weights. Using this lower bound, we get an oracle A such that $\Delta_k^{p,A} \not\subseteq \mathrm{PP}^{\Sigma_{k-2}^{p,A}}$.

2 A lower bound for perceptrons

We begin this section by defining the function f_k^m, first defined by Sipser [10], which can be computed by linear size circuits of depth k. Then we show the main theorem, which states that perceptrons of depth k with bounded fan-in that compute this function must be large. As a corollary we get that perceptrons of depth $k-1$ computing f_k^m must be large.

A perceptron is a circuit with a single threshold gate at the top. The threshold gate may have arbitrary weights and outputs 1 if its weighted sum of the input variables exceeds some threshold value. The inputs to the threshold gate are outputs of boolean circuits, the perceptron's *sub-circuits*. The sub-circuits consist of alternating levels of AND and OR gates; their top gate is either an AND gate for all the sub-circuits, or they may all be OR gates. Input variables may be negated, but otherwise no NOT gates may occur in the circuits. A general perceptron can be transformed into this form by at most doubling its size. A depth k perceptron has sub-circuits of depth $k-1$.

Definition 1. *The function f_k^m is a function of $m^{k-2}\frac{1}{2}(m\log m)^{1/4}\sqrt{\frac{1}{2}km\log m}$ variables. It is defined by a depth k circuit that has the form of a tree. At the leaves of the tree are unnegated variables.*

The root is an OR gate with fan-in $\frac{1}{2}(m\log m)^{1/4}$. Below are alternating levels of AND and OR gates with fan-in m. The bottom-most level has fan-in $\sqrt{\frac{1}{2}km\log m}$.

The logarithms in this definition, as well as in the rest of the paper, are base 2 logarithms.

The theorem is proved analogously to Håstad's proof that circuits of depth k are more powerful than circuits of depth $k-1$. A central part of the proofs is the Håstad switching lemma, which states that if we set a subset of the input variables to 0 or 1 randomly, we can switch the two lowest levels of AND and OR gates without increasing the circuit size very much. Then, two levels of the circuit can be collapsed into one.

The way that we set some input variables of a circuit C to 0 and 1 is through the use of *restrictions*. A restriction ρ is a mapping of the variables to the set $\{0,1,*\}$, where $*$ means that the variable remains unset. Let $C\lceil_\rho$ denote the circuit obtained by replacing each input variable x_i by $\rho(x_i)$. We use the same set of restrictions as in [6, Chapter 6]; they are defined below.

Definition 2. *Let q be a real number and $(B_i)_{i=1}^r$ a partition of the variables (that is, the B_i are disjoint and their union equals the set of all variables). Let $R_{q,B}^+$ be the probability space of restrictions which takes values as follows. For $\rho \in R_{q,B}^+$ and every B_i, $1 \le i \le r$, independently*

1. *With probability q let $s_i = *$ and else $s_i = 0$.*

2. *For every $x_k \in B_i$ let $\rho(x_k) = s_i$ with probability q and else $\rho(x_k) = 1$.*

Similarly a $R^-_{q,B}$ probability space of restrictions is defined by interchanging the roles played by 0 *and* 1.

Definition 3. *For a restriction $\rho \in R^+_{q,B}$, let $g(\rho)$ be a restriction defined as follows: for all B_i with $s_i = *$, $g(\rho)$ gives the value* 1 *to all variables given the value $*$ by ρ, except one to which it gives the value $*$. To make $g(\rho)$ deterministic, we assume that it gives the value $*$ to the variable with the highest index given the value $*$ by ρ.*

If $\rho \in R^-_{q,B}$, then $g(\rho)$ is defined similarly, but now takes the values 0 *and $*$.*

It follows from the definition of $g(\rho)$ that it never assigns values to the same variables as ρ, and for a circuit C, we denote $(C\lceil_\rho)\lceil_{g(\rho)}$ by $C\lceil_{\rho g(\rho)}$.

The variables can be partitioned according to what gates they occur at in the circuit that defines f^m_k. When the restrictions are used, the blocks B_i in the previous two definitions correspond to this partitioning.

We use the Håstad switching lemma for this set of restrictions [6, Lemma 6.3], and in the proof of the main theorem, we need an important property of the switching lemma, namely: after switching, the inputs accepted by the different ANDs form disjoint sets. This property was not explicitly stated by Håstad although it follows from his proof, and for this reason we include a sketch of the proof where we emphasize the "disjointness property" in the appendix.

Lemma 4. *Let G be an AND of ORs, all of size at most t, and ρ a random restriction from $R^+_{q,B}$ or $R^-_{q,B}$. Then the probability that $G\lceil_{\rho g(\rho)}$ cannot be written as an OR of ANDs all of size less than s, where the inputs accepted by the different ANDs form disjoint sets, is bounded by α^s where $\alpha = 6qt$.*

The same holds for converting an OR of ANDs to an AND of ORs.

The next lemma shows that applying a restriction to the defining circuit for f^m_k does not reduce it too much: with high probability the remaining circuit computes f^m_{k-1}. We fix the parameter q that is used for the restrictions to be $(\frac{2k}{m}\log m)^{1/2}$ for the rest of the paper.

Lemma 5 (Håstad [6, Lemma 6.8])**.** *If k is even the circuit that defines $f^m_k\lceil_{\rho g(\rho)}$ for a random $\rho \in R^+_{q,B}$, will contain the circuit that defines f^m_{k-1} with probability at least $2/3$, for all m such that $m/\log m \geq 100k$, $m > m_1$, where m_1 is some constant. The same is true for odd k when $R^+_{q,B}$ is replaced by $R^-_{q,B}$.*

Proof. Suppose k is even so that the restriction comes from $R^+_{q,B}$ and the lowest level of the defining circuit for f^m_k consists of AND gates; the other case is analogous.

The probability that one or more of the AND gates on the lowest level is reduced to the constant 1 by the restriction, or, equivalently, that one or more OR gates on the next lowest level is reduced to the constant 1 by the restriction, is bounded by $1/6$. This fact follows since the AND gate corresponding to the block B_i takes the value 1 with probability

$$\begin{aligned}(1-q)^{|B_i|} &< (1-(\tfrac{2k}{m}\log m)^{1/2})^{\sqrt{\frac{1}{2}km\log m}} \\ &= \left((1-(\tfrac{2k}{m}\log m)^{1/2})^{-\sqrt{\frac{m}{2k\log m}}}\right)^{-k\log m} \\ &< e^{-k\log m} < \frac{1}{6}m^{-k},\end{aligned}$$

and since there are less than m^k AND gates on the bottom-most level.

Now suppose we are in the (good) case that all AND gates take the values s_i. The probability that the number of remaining inputs to an OR gate (i.e., the number of AND gates not forced to 0) is less than $\sqrt{\frac{1}{2}km\log m}$ is at most $1/6m^{-k}$. This follows since the expected number of such gates is $qm = \sqrt{2km\log m}$, and using Chernoff's inequality we get that the probability of obtaining less than half the expected number is at least $e^{-qm/8} < 1/6m^{-k}$ for $m/\log m \geq 100k$. So, with probability $5/6$ none of the OR gates will have less than $\sqrt{\frac{1}{2}m\log m}$ inputs.

To sum up, we get a circuit that contains f_{k-1}^m with probability at least $2/3$. □

We use induction over the depth of the perceptron to prove the theorem. For the basis we need the following lemma.

Lemma 6. *The bottom fan-in of a depth two perceptron that computes f_2^m is at least $\frac{1}{2}(m\log m)^{1/4}$.*

Proof. The lemma follows from the "One-in-a-box" theorem by Minsky and Papert [9, Theorem 3.2] by substituting the variable m in their theorem by $\frac{1}{2}(m\log m)^{1/4}$. □

We are now ready to prove the main theorem.

Theorem 7. *There are no depth k perceptrons computing f_k^m with bottom fan-in $\frac{1}{\sqrt{3k}}(m\log m)^{1/4}$ and less than $2^{\frac{1}{\sqrt{3k}}(m\log m)^{1/4}}$ gates, not counting the gates on the lowest level, for $m \geq m_1$, where m_1 is some constant.*

Proof. We may assume that $m\log m \geq k^2$, since otherwise we have $2^{\frac{1}{\sqrt{3k}}(m\log m)^{1/4}} \leq 2^{1/\sqrt{3}}$.

The theorem is proved by induction over k. The basis $k = 2$ follows from Lemma 6. We first give an overview for the proof of the induction step, which is proved by contradiction.

We assume that there is a small-sized depth-k perceptron with bounded fan-in computing f_k^m. Then, we apply a random restriction to the inputs, and with high probability we get a perceptron that computes f_{k-1}^m. Also, because of the Håstad switching lemma, we can switch the two lower levels of ANDs and ORs in the perceptron without increasing the fan-in, and therefore collapse two levels to one and obtain a small-sized perceptron of depth $k-1$ and bounded fan-in computing f_{k-1}^m.

When $k = 3$ the procedure above does not work, since there are only two levels of AND and OR gates in the perceptron. However, we can deal with this as follows. Suppose that the lower level of the depth-three perceptron consists of OR gates (the other case is analogous). When switching an AND of ORs to an OR of ANDs, Lemma 4 says that the input sets accepted by the ANDs are mutually disjoint. Therefore, the output from the OR gate is always equal to the sum of the outputs of the AND gates, and we can thus substitute all OR gates that resulted after switching by summation gates, and thereafter collapse the two top-most levels.

We make the intuition formal by assuming that there is a depth-k perceptron P with bottom fan-in $\frac{1}{\sqrt{3k}}(m\log m)^{1/4}$ that has less than $2^{\frac{1}{\sqrt{3k}}(m\log m)^{1/4}}$ gates, not counting the gates on the lowest level, computing f_k^m.

Apply a random restriction from $R_{q,B}^+$ or $R_{q,B}^-$ depending on whether k is even or odd respectively. From Lemma 5 we have that with probability at least $2/3$, the perceptron P computes a function at least as hard as f_{k-1}^m. Note that this lemma requires $m/\log m \geq 100k$ which follows from $m\log m \geq k^2$ for large enough m.

Now we want to use Lemma 4 to show that all the sub-circuits on the two bottom levels can be switched. We know that the bottom fan-in is at most $t = \frac{1}{\sqrt{3k}}(m\log m)^{1/4}$, and to use the induction hypothesis we have to obtain a circuit with fan-in at most $s = \frac{1}{\sqrt{3(k-1)}}(m\log m)^{1/4}$. For each single

sub-circuit, the probability that this fails is at most $(6qt)^s$, which we multiply by the maximum number of such circuits to get a bound for the probability that this fails for at least one circuit:

$$2^{\frac{1}{\sqrt{3k}}(m\log m)^{1/4}}(6qt)^s = 2^{\frac{1}{\sqrt{3k}}(m\log m)^{1/4}}\left(6(\tfrac{2k}{m}\log m)^{1/2}\tfrac{1}{\sqrt{3k}}(m\log m)^{1/4}\right)^{\frac{1}{\sqrt{3(k-1)}}(m\log m)^{1/4}}$$
$$\leq \left(12(\tfrac{2k}{m}\log m)^{1/2}\tfrac{1}{\sqrt{3k}}(m\log m)^{1/4}\right)^{\frac{1}{\sqrt{3(k-1)}}(m\log m)^{1/4}} \leq 1/2,$$

where the last inequality holds since $m\log m \geq k^2$.

Note that the number of gates that are not on the bottom-most level does not increase when switching, which we need to use the induction hypothesis.

Thus, with probability at least $2/3$, $P\lceil_{\rho g(\rho)}$ computes a function at least as hard as f_{k-1}^m, and with probability at least $1/2$, $P\lceil_{\rho g(\rho)}$ is a function that does not compute f_{k-1}^m by the induction hypothesis. Therefore, both of these events must happen simultaneously with positive probability, a contradiction. □

Corollary 8. *There are no depth $k-1$ perceptrons that compute f_k^m with less than $2^{\frac{1}{\sqrt{3k}}(m\log m)^{1/4}}$ gates, for $m \geq m_1$, for some constant m_1.*

Proof. Assume there is a perceptron such that the corollary does not hold. Then, adding a gate with fan-in one to all the inputs yields a depth k perceptron that does not exist according to Theorem 7. □

Corollary 9. *There are functions computable by linear size circuits of depth k that require super-polynomial size perceptrons of depth $k-1$ if $3 \leq k < \frac{\log n}{6\log\log n}$, where n is the number of input variables.*

Proof. From Corollary 8 we have that if $\frac{1}{\sqrt{3k}}(m\log m)^{1/4} \in \omega(\log n)$ then depth $k-1$ perceptrons computing f_k^m require super-polynomial size. We have $n > m^{k-2}$ and thus $k-2 < \log n/\log m$ and $\log m/\log n < 1$. For $m\log m \geq k^2$ we have $n < m^k$ and thus $k > \log n/\log m$. We get

$$\sqrt{3k} < \sqrt{3\frac{\log n}{\log m}+6} = \sqrt{\frac{\log n}{\log m}\left(3+6\frac{\log m}{\log n}\right)} < 3\sqrt{\frac{\log n}{\log m}}$$

so that

$$\frac{1}{\sqrt{3k}}(m\log m)^{1/4} > \left(\frac{m\log^3 m}{\log^2 n}\right)^{1/4},$$

which is $\omega(\log n)$ if $m\log^3 m \in \omega(\log^6 n)$. This holds if $k < \frac{\log n}{6\log\log n}$, since we then have $(\log^6 n)^k < n < m^k$ so that $m > \log^6 n$. □

3 Oracle separation

In this section we show how the lower bound in Theorem 7 implies the existence of an oracle that separates the different levels in the $\mathrm{PP}^{\mathrm{PH}}$ hierarchy.

An oracle A is a fixed set of strings called an *oracle set*. Let y_z^A denote the characteristic function for the oracle A which is 1 if the string z is in A.

An *alternating Turing machine* is a non-deterministic Turing machine whose states are marked by $\wedge$, $\vee$, 0, or 1. States marked $\wedge$ and $\vee$ have at most two next configurations, and states marked 0 and 1 are the halting states, in which the machine rejects or accepts, respectively. Let a $\Sigma_d^{p,A}$ machine denote an alternating Turing machine with access to the oracle A. Such a machine has an additional oracle

query tape; when the query tape contains the string z, the machine can enter a special oracle query state to compute y_z^A in one time step.

Given an input value x to a $\Sigma_d^{p,A}$ machine, the result of the computation is determined by evaluating the computation tree in the natural way. The computation tree is defined as follows. Every possible machine configuration is represented by a node, which is labeled by $\wedge$, $\vee$, 0, or 1, depending on the marking of the corresponding state. A node is the parent to those nodes that represent the possible next configurations; thus the nodes representing halting configurations are the leaves of the tree. The maximum number of blocks of consecutive states labeled by $\wedge$ or $\vee$ on a path from the root to a leaf is d, and the block of states closest to the root is an $\vee$ block.

A $\mathrm{PP}^{\Sigma_{d-1}^{p,A}}$ machine is defined like a $\Sigma_d^{p,A}$ machine, but where the machine starts its computation in a state marked by $+$ (a counting state). In the computation tree, the block closest to the root is labeled by $+$, and the machine accepts if the evaluation of the tree exceeds a threshold value.

The complexity classes $\Sigma_d^{p,A}$ and $\mathrm{PP}^{\Sigma_{d-1}^{p,A}}$ contain exactly those languages that can be decided by $\Sigma_d^{p,A}$ and $\mathrm{PP}^{\Sigma_{d-1}^{p,A}}$ machines in polynomial time, respectively.

Given an alternating Turing machine, it can be converted to a machine that makes all oracle queries at the end of the computation with only a polynomial increase in execution time and no extra alternations. We will assume that all machines have this form for the rest of the paper. The conversion is done as follows. Oracle queries are replaced by non-deterministic guesses for the oracle answer. Along one computation branch the machine assumes the answer 0, and along the other it assumes the answer 1; it remembers the question, the guess for the answer, and the marking of the original query state for the rest of the computation. A halting state is replaced by a number of states that verify the guesses made in the computation. If all the guesses were correct, the computation branch accepts or rejects according to the marking of the original halting state. Otherwise, the machine makes sure that the computation branch does not affect the result of the computation by accepting if the first incorrect guess was made in an $\wedge$ state, and rejecting otherwise.

The following relation between alternating oracle Turing machines and perceptrons is similar to for example Lemma 2.1 in [8] and Corollary 2.2 in [3].

Lemma 10. *Let M^A be a $\mathrm{PP}^{\Sigma_{d-1}^{p,A}}$ oracle machine which runs in time t on input x. Then there is a depth $d+1$ perceptron P with unit weights and bottom fan-in t which has a subset of the y_z^A as inputs such that for every oracle A, M^A accepts x precisely when P outputs 1 on inputs y_z^A. This perceptron has at most 2^t gates, not counting gates on the bottom-most level.*

Proof. For a fixed x, write down the computation tree for $M^A(x)$ for some oracle A. On a path from the root of the tree to a leaf, let the first node where an oracle query occurs (if one exists) be called a *boundary node*. Boundary nodes mark where in the computation that the machine starts verifying its guesses, and since this is the last part of a computation branch, and since it is deterministic, there is exactly one leaf under each boundary node.

When varying the oracle A we get different computation trees. However, the part of the computation trees that are above the boundary nodes remain the same regardless of the oracle. The part of the computation trees below a boundary node accepts or rejects depending on the oracle. However, note that the only oracle queries that may occur at and below the boundary node are the ones for which guesses have been made above the boundary node; their number is bounded by t.

We now construct a perceptron from the computation tree of $M^A(x)$. Every boundary node in the computation tree is replaced by a DNF or CNF formula, depending on if the boundary node is labeled by $\vee$ or $\wedge$, respectively. In the case of boundary nodes labeled by $\vee$, we make a conjunction for each possible set of oracle answers that makes the computation branch accept, and combine them into a DNF formula. In the case of boundary nodes labeled $\wedge$, make a conjunction for each possible set

of oracle answers that makes the computation branch reject, and combine them into a DNF formula. Taking the negation of this formula and applying DeMorgan's law yields a CNF formula. Finally, the resulting tree is collapsed to yield a perceptron of depth $d+1$ and bottom fan-in t.

The depth of the original computation tree is at most t, and hence its size at most 2^t. Since new gates from the DNF and CNF formulas only appear on the bottom-most level, the maximum number of gates on higher levels in the resulting perceptron is at most 2^t. □

Theorem 11. *There exists an oracle A such that, for all d, there is a language $L_d(A)$ in $\Sigma_d^{p,A}$ which is not recognizable by any $PP^{\Sigma_{d-2}^{p,A}}$ machine, i.e., $\Sigma_d^{p,A} \not\subseteq PP^{\Sigma_{d-2}^{p,A}}$.*

Proof. The intuition for the proof is that a $\mathrm{PP}^{\Sigma_{d-2}^{p,A}}$ machine corresponds to a depth d perceptron with inputs y_z^A for each input x. Theorem 7 from the last section is used to show that such a perceptron is too small for computing the function f_d^m of the oracle bits for some m. We define a language $L_d(A)$ that depends on A in a way that for a machine to decide the language properly for all oracles would require the corresponding perceptron to compute f_d^m; thus we can choose an oracle such that the machine makes an error.

Let $x = x_1 x_2 \cdots x_n$ and

$$L_d(B) = \{1^n \mid \exists x_1, x_2, \ldots, x_{n/d} \forall x_{n/d+1}, \ldots, x_{2n/d} \cdots Q x_{n(d-1)/d+1}, \ldots, x_n : x \in B\}.$$

Now, a $\mathrm{PP}^{\Sigma_{d-2}^{p,B}}$ machine that decides if $1^n \in L_d(B)$ has a corresponding perceptron of depth d that evaluates a function h_d^n in the variables y_z^B for $|z| = n$, as exemplified in Figure 1.

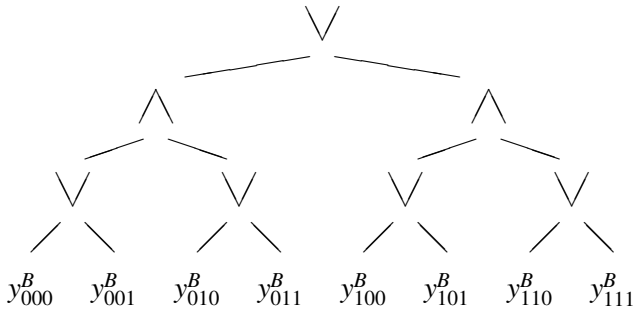

Figure 1: To check if $1^3 \in L_3(B)$ corresponds to evaluating the function h_3^3 given by this circuit.

The function h_d^n resembles $f_d^{2^{n/d}}$, the only difference being the fan-in on the top and bottom level which for h_d^n is $2^{n/d}$ while it is lower for $f_d^{2^{n/d}}$. This means that h_d^n contains $f_d^{2^{n/d}}$, and is therefore at least as hard to compute.

By the construction of $L_d(B)$ it is clear that $L_d(B) \in \Sigma_d^{p,B}$, and we now construct an oracle B such that $L_d(B)$ cannot be decided by a $\mathrm{PP}^{\Sigma_{d-2}^{p,B}}$ machine for any d.

Let M_i^B for $i = 1, 2, \ldots$ be an enumeration of $\mathrm{PP}^{\Sigma_{d-2}^{p,B}}$ machines for all constants d. The oracle will be built in rounds, and in round i we will make sure that the machine M_i makes an error for some input.

In round i we do the following: Suppose M_i^B is a $\mathrm{PP}^{\Sigma_{d-2}^{p,B}}$ machine which runs in time cn^c (observe that c and d depend on i). We want to fix some of the y_z^B such that M_i^B makes an error for at least one of the strings in $L_d(B)$. More precisely, we make M_i^B fail on the string 1^{n_i}, where n_i is chosen such that both the following statements are satisfied, which they are for large enough n_i.

1. None of the y_z^B with $|z| = n_i$ have previously been set.

2. $\frac{1}{\sqrt{3d}}\left(2^{n_i/d}n_i/d\right)^{1/4} > cn_i^c$.

The first requirement makes sure that none of the strings previously set in the oracle interfere with the current machine on input 1^{n_i}. For any oracle query y_z^B with $|z| \neq n_i$ that M_i^B may make, we substitute the correct value for those that are already determined, and fix previously undetermined variables to some arbitrary value.

From Lemma 10 there is a perceptron of depth d, bottom fan-in cn^c, and with at most 2^{cn^c} gates (excluding those on the bottom-most level) with a subset of the y_z^B as inputs that outputs 1 for the oracles that make M_i^B accept the string 1^{n_i}. This perceptron is, due to Theorem 7, not powerful enough to compute $h_d^{n_i}$ and is thus unable to determine if the string 1^{n_i} is in $L_d(B)$ for all B. It is therefore possible to set the y_z^B of length n_i such that M_i^B makes an error on 1^{n_i}. □

4 Improving the oracle separation

Beigel [1] obtained the oracle separation $\mathrm{P}^{\mathrm{NP}} \not\subseteq \mathrm{PP}$. To do this he introduced the language ODD-MAX-BIT and proved a relation between the bottom fan-in, the maximum weight, and the size for perceptrons deciding it.

Definition 12 (Beigel [1, Definition 2])**.** *ODD-MAX-BIT is the set of all strings over $\{0,1\}^*$ whose rightmost 1 is in an odd-numbered position, i.e., the set of strings of the form $x10^k$ where the length of x is even.*

We use this function to define a more complex one that is suited for obtaining a lower bound for perceptrons of depth k. We first slightly change the definition of the function f_k^m from the previous section.

Definition 13. *The function f_k^m is a function of $m^{k-1}\sqrt{\frac{1}{2}km\log m}$ variables. It is defined by a depth k circuit that has the form of a tree. At the leaves of the tree are unnegated variables.*

The bottom-most level has fan-in $\sqrt{\frac{1}{2}km\log m}$. The rest of the levels, including the root for $k > 1$, all have fan-in m. The root is an AND gate, and below are alternating levels of OR and AND gates.

The function that we obtain a lower bound for is g_k^m, defined below.

Definition 14. *For $k = 2$, g_2^m computes the ODD-MAX-BIT function of $\frac{1}{6}\sqrt{m\log m}$ variables.*

For $k > 2$, g_k^m is a function of $m^{k-2}\sqrt{\frac{1}{2}(k-2)m\log m}$ variables. It computes the ODD-MAX-BIT function of the outputs of m disjoint f_{k-2}^m functions.

When applying a random restriction to a circuit computing g_k^m, as is the case with f_k^m, with high probability we get a circuit that computes g_{k-1}^m. As before, fix the parameter q that is used for the restrictions to be $(\frac{2k}{m}\log m)^{1/2}$.

Lemma 15. *If k is odd the circuit that defines $g_k^m\lceil_{\rho g(\rho)}$ for a random $\rho \in R_{q,B}^+$ will contain the circuit that defines g_{k-1}^m with probability at least $2/3$, for all m such that $m/\log m \geq 100k$, $m > m_1$, where m_1 is some constant. The same is true for even k when $R_{q,B}^+$ is replaced by $R_{q,B}^-$.*

Proof. When $k > 3$, the proof works as for Lemma 5. The case $k = 3$ is now special and let's consider this case; the circuit computes the ODD-MAX-BIT function of m AND gates, each with fan-in $\sqrt{\frac{1}{2}m\log m}$. As in the proof of Lemma 5, with probability at least $5/6$, none of the AND gates is reduced to the constant 1, and they thus take the values s_i.

When all m AND gates take the values s_i (which is 0 or $*$) by the restriction, we show that with high probability the remaining circuit contains a circuit that computes the ODD-MAX-BIT function of $\frac{1}{6}\sqrt{m\log m}$ variables.

Divide the m inputs to the ODD-MAX-BIT function into $\sqrt{m\log m}$ blocks D_i of size $|D_i| = \sqrt{m/\log m}$. If, in a block D_i, there is a $*$ in both an even- and an odd-numbered position, then we can surely use one of these bits. The probability that a block does not get any $*$ in an even numbered position is $(1-q)^{|D_i|/2}$; the same holds for odd-numbered positions. Thus, with probability at least

$$1-2(1-q)^{|D_i|/2} = 1-2\left(\left(1-\sqrt{\frac{6\log m}{m}}\right)^{-\sqrt{\frac{m}{6\log m}}}\right)^{-\sqrt{6}/2} > 1-2e^{-\sqrt{6}/2} > 1/3$$

we can pick an input from a given block. Since there are $\sqrt{m\log m}$ blocks, the expected number of blocks from which we can pick an input is at least $\frac{1}{3}\sqrt{m\log m}$, and using Chernoff's inequality, we get that less than $\frac{1}{6}\sqrt{m\log m}$ blocks have the property with probability $e^{-\sqrt{m\log m}/24} < 1/6$. □

A depth two perceptron that contains no AND gates with negated literals and no identical AND gates is said to be in *positive normal form* by Minsky and Papert [9]. Beigel called this *clean form*, and he formulated the following lemma, which uses the construction from [9, page 33].

Lemma 16 (Beigel [1, Lemma 1])**.** *If f is computed by a perceptron with size s, maximum weight w, and order d, then f is computed by a perceptron in clean form with size $2^d s$, weight sw, and order d.*

Lemma 17. *There are no depth two perceptrons with unit weights that compute g_2^m with bottom fan-in $m^{1/6}$ and less than $2^{5m^{1/6}}$ gates, for m larger than some constant.*

Proof. A lemma due to Beigel [1, Lemma 5] gives the following relation between the size s, maximum weight w, and order d for depth two perceptrons on clean form computing ODD-MAX-BIT of N input bits:

$$w \geq \frac{1}{s} 2^{\left\lfloor \frac{N-1}{2\lceil d^2/(\sqrt{87}-9)\rceil} \right\rfloor}.$$

Suppose there is a depth two perceptron with unit weights that computes g_2^m that has bottom fan-in (order) $m^{1/6}$ and less than $2^{5m^{1/6}}$ gates. Then, due to Lemma 16, there is a perceptron in clean form with fan-in $m^{1/6}$, size $2^{6m^{1/6}}$ and weights bounded by $2^{5m^{1/6}}$ computing ODD-MAX-BIT of $\frac{1}{6}\sqrt{m\log m}$ variables.

If we put these values into Beigel's formula above, we get a contradiction for large enough m. □

Theorem 18. *For $k \geq 3$, there are no depth k perceptrons computing g_k^m with unit weights, bottom fan-in $\frac{1}{\sqrt{k}}m^{1/6}$, and less than $2^{6\frac{1}{\sqrt{k}}m^{1/6}}$ gates, not counting the gates on the lowest level, for m larger than some constant.*

When $k = 2$ we have a somewhat stronger result. The bound for the number of gates holds when counting all gates in the circuit.

Proof. We may assume that $m \geq k^3$, since otherwise we have $2^{6\frac{1}{\sqrt{k}}m^{1/6}} \leq 2^6$. The basis for the induction, $k = 2$, follows from Lemma 17.

Assume there is a depth-k perceptron P with bottom fan-in $\frac{1}{\sqrt{k}}m^{1/6}$ that has less than $2^{6\frac{1}{\sqrt{k}}m^{1/6}}$ gates computing g_k^m.

To be able to apply Lemma 15, choose one of $R_{q,B}^+$ or $R_{q,B}^-$ and apply a random restriction to the perceptron. We have that with probability at least $2/3$, the perceptron P computes a function at least as hard as g_{k-1}^m.

The bottom fan-in is at most $t = \frac{1}{\sqrt{k}}m^{1/6}$, and to use the induction hypothesis we have to obtain a circuit with fan-in at most $s = \frac{1}{\sqrt{(k-1)}}m^{1/6}$. For each single gate on the next to the lowest level, the probability that this fails is at most $(6qt)^s$ due to Lemma 4, and we multiply this by the maximum number of switchings to get a bound for the probability that the conversion fails for at least one circuit:

$$\begin{aligned} 2^{6\frac{1}{\sqrt{k}}m^{1/6}}(6qt)^s &= 64^{\frac{1}{\sqrt{k}}m^{1/6}}\left(6(\tfrac{2k}{m}\log m)^{1/2}\tfrac{1}{\sqrt{k}}m^{1/6}\right)^{\frac{1}{\sqrt{(k-1)}}m^{1/6}} \\ &\leq \left(6\cdot 64(\tfrac{2k}{m}\log m)^{1/2}\tfrac{1}{\sqrt{k}}m^{1/6}\right)^{\frac{1}{\sqrt{(k-1)}}m^{1/6}} \leq 1/2, \end{aligned}$$

where the last inequality holds since $m \geq k^3$.

Going from depth 3 to depth 2 is a special case, since we need to bound the number of gates in the entire depth 2 circuit. Suppose without loss of generality that the lowest level of the depth 3 perceptron consists of OR gates (if it consists of AND gates we can negate the perceptron's weights and use deMorgan's law). Then, the depth 2 perceptron that results after switching has AND gates on the lowest level, and the fan-in of these gates is bounded by s so that each of them accepts at least a fraction 2^{-s} of the inputs. We know from Lemma 4 that all the AND gates accept disjoint inputs so we get that the maximum number of AND gates that results from each switching is bounded by 2^s. Thus, the total number of gates in the resulting depth 2 circuit is at most

$$2^s 2^{6\frac{1}{\sqrt{k}}m^{1/6}} = 2^{\frac{1}{\sqrt{2}}m^{1/6}}2^{\frac{6}{\sqrt{3}}m^{1/6}} < 2^{\frac{6}{\sqrt{2}}m^{1/6}}.$$

□

Theorem 19. *There exists an oracle A such that, for all d, there is a language $L_d(A)$ in $\Delta_d^{p,A}$ which is not recognizable by any $PP^{\Sigma_{d-2}^{p,A}}$ machine, i.e., $\Delta_d^{p,A} \not\subseteq PP^{\Sigma_{d-2}^{p,A}}$.*

Proof. The proof is analogous to the proof of Theorem 11, but we change the language to one that can be decided by a $\Delta_d^{p,A}$ machine. Due to Theorem 18, however, the perceptron corresponding to a $\mathrm{PP}^{\Sigma_{d-2}^{p,A}}$ machine deciding the language is too small for computing the function g_d^m of the oracle bits for some m.

Let

$$L'_d(B) = \{y \mid \forall x_1, x_2, \ldots, x_{|y|} \exists x_{|y|+1}, \ldots, x_{2|y|} \cdots Q x_{(d-1)|y|+1}, \ldots, x_{d|y|} : yx \in B\},$$

where $x_1, x_2, \ldots$ denote the individual bits of x. We then define

$$L_d(B) = \{1^n \mid \max(L'_{d-2}(B) \cap \{0,1\}^{n/(d-1)}) \text{ ends in a } 1\}.$$

Now, a $\mathrm{PP}^{\Sigma_{d-2}^{p,B}}$ machine that decides if $1^n \in L_d(B)$ has a corresponding perceptron of depth d that evaluates a function h_d^n in the variables y_z^B for $|z| = n$. The function h_d^n is at least as hard to compute as $g_d^{2^{n/(d-1)}}$.

The language $L_d(B)$ is in $\Delta_d^{p,B}$ since it can be decided by a P^{NP} machine that uses a $\Sigma_{d-2}^{p,B}$ machine as an oracle.

The construction of an oracle B such that $L_d(B)$ cannot be decided by a $\mathrm{PP}^{\Sigma_{d-2}^{p,B}}$ machine for any d is now as in the proof of Theorem 11, the only difference being that n_i is chosen such that

$$\frac{6}{\sqrt{d}}\left(2^{n_i/(d-1)}\right)^{1/6} > cn_i^c$$

in each round. □

Acknowledgments

We thank Mikael Goldmann and Johan Håstad for helpful discussions.

References

[1] Richard Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4(4):339–349, 1994.

[2] Bin Fu. Separating PH from PP by relativization. *Acta Mathematica Sinica (New Series)*, 8(3):329–336, 1992.

[3] Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[4] Frederic Green. An oracle separating $\oplus$P from PP$^{\text{PH}}$. *Information Processing Letters*, 37:149–153, 1991.

[5] Frederic Green. A lower bound for monotone perceptrons. *Mathematical Systems Theory*, 28:283–298, 1995.

[6] Johan Håstad. *Computational Limitations for Small-Depth Circuits*. ACM doctoral dissertation awards. MIT Press, Cambridge, Massachusetts, 1987.

[7] Johan Håstad. Almost optimal lower bounds for small depth circuits. In Silvio Micali, editor, *Randomness and Computation*, volume 5 of *Advances in Computing Research*, pages 143–170. JAI Press Inc, 1989.

[8] Ker-I Ko. Relativized polynomial time hierarchies having exactly k levels. *SIAM Journal of Computing*, 18(2):392–408, 1989.

[9] Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, Cambridge, Massachusetts, expanded edition, 1988.

[10] Michael Sipser. Borel sets and circuit complexity. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pages 61–69, 1983.

Appendix

Proof of Lemma 4. Let $\text{AND}(G\lceil_{\rho g(\rho)}) \geq s$ denote that $G\lceil_{\rho g(\rho)}$ can't be written as an OR of ANDs that accept disjoint sets, all ANDs having size less than s.

We use induction to prove the lemma, and in fact, we prove that if $G = \wedge_{i=1}^{w} G_i$, where G_i are ORs of fan-in at most t, then

$$\text{P}[\text{AND}(G\lceil_{\rho g(\rho)}) \geq s \mid F\lceil_{\rho} \equiv 1] \leq \alpha^s,$$

for an arbitrary F.

The basis $w = 0$ is obvious. For the induction step first rewrite

$$\begin{aligned}\text{P}[\text{AND}(G\lceil_{\rho g(\rho)}) \geq s \mid F\lceil_{\rho} \equiv 1] \leq \max(&\text{P}[\text{AND}(G\lceil_{\rho g(\rho)}) \geq s \mid F\lceil_{\rho} \equiv 1 \wedge G_1\lceil_{\rho} \equiv 1],\\ &\text{P}[\text{AND}(G\lceil_{\rho g(\rho)}) \geq s \mid F\lceil_{\rho} \equiv 1 \wedge G_1\lceil_{\rho} \not\equiv 1]).\end{aligned}$$

The first term is taken care of by the induction hypothesis, and the rest of the proof deals with the estimate of the second term.

We denote the set of variables occuring in G_1 by T and we have that $|T| \leq t$. If $G\lceil_\rho$ is identically 0, $G\lceil_{\rho g(\rho)}$ does not require long ANDs, so we may assume that it is not 0, and therefore that some of the variables in T must be given the value $*$ by ρ.

A block B is *exposed* if there is a variable $x_i \in B \cap T$ and $\rho(x_i) = *$. If the variables in T belong to the r different blocks $B_1, \ldots, B_r$, we know that some of these blocks are exposed. Let $\exp(Y)$, $Y \subseteq \{1, \ldots, r\}$, denote that the blocks indexed by Y are the ones that are exposed.

We get

$$\begin{aligned}
&\mathrm{P}[\mathrm{AND}(G\lceil_{\rho g(\rho)}) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1] \\
&\leq \sum_{\emptyset \neq Y \subseteq \{1,\ldots,r\}} \mathrm{P}[\exp(Y) \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1] \\
&\qquad \cdot \mathrm{P}[\mathrm{AND}(G\lceil_{\rho g(\rho)}) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1 \wedge \exp(Y)],
\end{aligned}$$

and the first factor is bounded by $(2q)^{|Y|}$ by Lemma 6.6 in [6]; it remains to estimate the second factor.

Let Y^* denote the set of variables that remain in the exposed blocks after the restriction $\rho g(\rho)$ when the blocks indexed by Y are exposed. We rewrite

$$G = \bigvee_{\sigma \in \{0,1\}^{|Y|}} G\lceil_{Y^*=\sigma} \wedge Q(Y^*, \sigma),$$

where $Q(Y^*, \sigma)$ denotes a predicate that is true when the variables in Y take the values of σ; it can be written as an AND of size $|Y|$.

Partition $\rho = \rho_1\rho_2$ into ρ_1 that is the restriction on the exposed blocks, and ρ_2 that is the restriction on the blocks that are not exposed.

We get

$$\begin{aligned}
&\mathrm{P}_\rho[\mathrm{AND}(G\lceil_{\rho g(\rho)}) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1 \wedge \exp(Y)] \\
&\quad = \mathrm{P}_\rho[\mathrm{AND}(G\lceil_{\rho g(\rho)}) \geq s \mid F'\lceil_{\rho_2} \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1 \wedge \exp(Y)] \\
&\quad = \mathrm{P}_\rho\Big[\mathrm{AND}\Big(\bigvee_{\sigma \in \{0,1\}^{|Y|}} (G\lceil_{Y^*=\sigma})\lceil_{\rho g(\rho)} \wedge Q(Y^*, \sigma)\Big) \geq s \,\Big|\, F'\lceil_{\rho_2} \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1 \wedge \exp(Y)\Big] \\
&\quad \leq \sum_{\sigma \in \{0,1\}^{|Y|}} \mathrm{P}_\rho[\mathrm{AND}((G\lceil_{Y^*=\sigma})\lceil_{\rho g(\rho)}) \geq (s - |Y|) \mid F'\lceil_{\rho_2} \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1 \wedge \exp(Y)] \\
&\quad \leq \sum_{\sigma \in \{0,1\}^{|Y|}} \max_{\rho_1} \mathrm{P}_{\rho_2}[\mathrm{AND}((G\lceil_{Y^*=\sigma}\lceil_{\rho_1 g(\rho_1)})\lceil_{\rho_2 g(\rho_2)}) \geq (s - |Y|) \mid F'\lceil_{\rho_2} \equiv 1] \\
&\quad \leq 2^{|Y|}\alpha^{s-|Y|}.
\end{aligned}$$

For the first equality, note that the condition $G_1\lceil_\rho \not\equiv 1$ is equivalent to $G_1\lceil_{\rho_1} \not\equiv 1 \wedge G_1\lceil_{\rho_2} \not\equiv 1$. Since ρ_2 assigns only the values 0 and 1 to variables in T, the second of these requirements can be combined with $(F\lceil_{\rho_1})\lceil_{\rho_2} \equiv 1$, and we write this as $F'\lceil_{\rho_2} \equiv 1$.

In the second equality, we don't have to apply the restriction to $Q(Y^*, \sigma)$, since the variables in Y^* are not affected by it anyway.

The first inequality is obtained by noting that if for all σ, $(G\lceil_{Y^*=\sigma})\lceil_{\rho g(\rho)}$ can be written as an OR of ANDs, all of length less than $s - |Y|$, and all of which accept disjoint input sets, then $G\lceil_{\rho g(\rho)}$ can be written as an OR of ANDs, all of length at most s, in a way that they all accept disjoint input sets.

In the second inequality we use that for boolean predicates R and S we have $\mathrm{P}_{\rho_1,\rho_2}[R(\rho_1,\rho_2) \mid S(\rho_1)] \leq \max_{\rho_1:S(\rho_1)} \mathrm{P}_{\rho_2}[R(\rho_1,\rho_2)]$ to get rid of the two last conditions.

We are finally in a position to use the induction hypothesis, which is done in the last inequality.

We now evaluate the sum to get

$$\begin{aligned}\sum_{\emptyset \neq Y \subseteq \{1,\ldots,r\}} (2q)^{|Y|} 2^{|Y|} \alpha^{s-|Y|} &= \alpha^s \sum_{i=1}^{r} \binom{r}{i} \left(\frac{4q}{\alpha}\right)^i \\ &= \alpha^s((1+4q/\alpha)^r - 1) \leq \alpha^s((1+2/(3t))^t - 1) \\ &\leq \alpha^s((e^{2/(3t)})^t - 1) \leq \alpha^s.\end{aligned}$$

□