# Efficiently finding small representations for LTFs

Johan Håstad
KTH- Royal Institute of Technology

December 8, 2025

**Abstract**

It is well known that any Linear Threshold Function, $f$, on $\{0,1\}^n$ has a representation with integer coefficients with $O(n \log n)$ bits. We study the problem of finding a small representation in polynomial time. Given a representation of $f$ with arbitrary size coefficients, we give a polynomial time algorithm that finds a representation with integer weights with $O(n^2)$ bits.

## 1 Introduction

Linear Threshold Functions (LTFs) give a simple but interesting class of Boolean functions. In this paper we use $\{-1,1\}$ to represent bits and an $n$-input LTF is defined by a set of integer weight $(w_i)_{i=0}^n$ where the value on $x \in \{-1,1\}^n$ is

$$f_w(x) = \mathrm{sign}(w_0 + \sum_{i=1}^n w_i x_i).$$

It is convenient to extend $x$ by adding a first coordinate, $x_0$, that is always 1 and consider $f_w$ as an odd function of $n+1$ variables given by the sign of the inner product $(w,x)$. We require that $(w,x) \neq 0$ for all $x \in \{-1,1\}^{n+1}$.

It is well known and has been proved many times, one early source being [Mur71], that it is sufficient to use integers, $w_i$, with $O(n \log n)$ bits. It turns out that this bound is sharp as there are examples [Hå94, AV97] of LTFs that require such large weights. The purpose of the current paper is to discuss the problem of finding weights of limited size efficiently.

If we are given the input, $f$, as a truth-table of size $2^n$, then the procedure of [Mur71] to find $w_i$ can be carried out in polynomial time, (which is $2^{O(n)}$ as the input is big). We do not give the details but the key subroutine is to set up a linear program containing the inequalities

$$w_0 + \sum_{i=1}^n x_i w_i \geq 1$$

when $f(x) = 1$ and

$$w_0 + \sum_{i=1}^{n} x_i w_i \leq -1$$

when $f(x) = -1$. Some modifications of the obtained solution can used to find an integer representation with small coefficients. This solves the problem of a polynomial time algorithm when the input size is $2^n$.

In our eyes a more realistic and interesting setting is when we are given some representation of $f$ with large weights and we want to find a representation with weights of bounded size. In this situation we are looking for algorithms that run in time polynomial in the number of bits in the given weights. The main result of this paper is such an algorithm that finds weights with $O(n^2)$ bits. Let us briefly describe the intuition of our algorithm.

If the weights are much larger than $2^n$ and random looking, then $(w, x)$ has a large absolute value for any $x$ and hence we can simply drop the least significant bit of each $w_i$, decreasing all weights by a factor of 2. It might be, however, a few $x$ such that $(w, x)$ has a small absolute value and we set up a lattice to identify a linear space, $W^1$, containing these $x$. We then show that it is possible to shrink $w$ by subtracting a vector orthogonal to $W^1$.

## 2 Some preliminaries

In this paper we use lattices and let us recall some basic facts and fix notation. For a more complete treatment we refer to a textbook such as [MG02].

A lattice, $L$, is given by a basis $(b_i)_{i=1}^n$ of linearly independent vectors in $\mathbb{R}^m$. $L$ consists of all points of the form $\sum_{i=1}^n a_i b_i$ where $a_i \in \mathbb{Z}$. It is often convenient to write a basis as the rows of an $n \times m$ matrix $B$.

The determinant of a lattice is a measure of its density. If $m = n$ it is simply $|\det(B)|$ while if $n < m$ it is more convenient to define it as $\det(BB^T)^{1/2}$.

A very important problem is, given a basis, to find a short vector in the corresponding lattice. It turns out that it is NP-hard (under randomized reductions) to find the shortest vector [Ajt98] and even to approximate its length [Kho05]. There are some positive algorithmic results and a classical polynomial time algorithm by Lovász [LLL82] finds a vector in the lattice whose length is within a factor $2^{(n-1)/2}$ of the shortest vector. This bound has been improved by Schnorr [Sch87] by a slightly more complicated algorithm. Using the algorithm of Schnorr would improve the bounds of the current paper somewhat, but as the algorithm of Lovász gives very easy to state bounds and is well known, we use this algorithm and let us recall some facts. Given the list of authors of the paper where it was published we call the algorithm "LLL".

For any basis $B$ of a lattice $L$, let $b_i^*$ be the component of $b_i$ which is orthogonal to $b_1, b_2, \ldots b_{i-1}$. LLL produces, in polynomial time, a basis such that $|b_i^*| \geq (1/2)^{1/2} |b_{i-1}^*|$ for $i \geq 2$. It is not difficult to see that that any vector in $L$ is of length at least $\min_i |b_i^*|$ and we conclude that the length of $b_1$ is within a factor $2^{(n-1)/2}$ of the shortest vector in $L$. The basis produced by LLL is also "size reduced" which is to say that

$$|(b_i, b_j*)| \leq \frac{1}{2}\|b_j^*\|^2$$

for any $i > j$.

Let us also note that it is not difficult to see that for any basis the determinant of $L$ equals $\prod_{i=1}^n \|b_i^*\|$. We also have the following standard lemma.

**Lemma 2.1** *Given a basis $b_i$ of $L$ and a vector $v$ in the linear span of this basis. Then it is possible to efficiently find $x \in L$ such that*

$$\|v - x\| \leq \frac{1}{2}\sum_{i=1}^{n} \|b_i^*\|.$$

**Proof:** (Sketch) We write $x = \sum_{i=1}^n a_i b_i$ and determine $a_i \in \mathbb{Z}$ for $i = n, n-1, \ldots 1$ such that $|(v - x, b_i^*)| \leq \frac{1}{2}\|b_i^*\|^2$. ■

To make formulas more compact, we assume in all calculations that $n$ is sufficiently large. This implies in particular that any polynomial function in $n$ is smaller than any exponential function of $n$. This convention implies that our stated bounds are only proved for "sufficiently large $n$". We do not explicitly state this in each lemma and we use the convention without mentioning it in calculations.

## 3 The algorithm

We are given integer weights $w_i$ with $\|w\| = 2^\ell \geq 2^{4n^2}$. Let $\Lambda_m$ be a lattice of dimension $n + 1$ in $n + 2$ dimensions given by the rows of the matrix

$$\begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & 2^{-m}w_0 \\ 0 & 1 & 0 & \ldots & 0 & 2^{-m}w_1 \\ 0 & 0 & 1 & \ldots & 0 & 2^{-m}w_2 \\ \vdots & \vdots & \vdots & \ldots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 2^{-m}w_n \end{pmatrix}.$$

It is not difficult to see that the determinant of $\Lambda_m$ is roughly $2^{-m}\|w\|$ and, even if this is all we need, the determinant is not hard to compute exactly.

**Lemma 3.1** *The determinant of $\Lambda_m$ is $(1 + 2^{-2m}\|w\|^2)^{1/2}$.*

**Proof:** Extend $\Lambda_m$ to an $n+2$ dimensional lattice, $L$, by adding a basis vector

$$v' = (2^{-m}w_0, 2^{-m}w_1, 2^{-m}w_2 \ldots, 2^{-m}w_n, -1).$$

As $v'$ is orthogonal to all vectors in $\Lambda_m$ we have

$$\det(L) = \det(\Lambda_m)\|v'\| = \det(\Lambda_m)(1 + 2^{-2m}\|w\|^2)^{1/2}.$$

The determinant of $L$ is easy to calculate as it is the determinant of a very sparse $(n+2) \times (n+2)$ matrix. Indeed, expanding it along the last column shows that

$$\det(L) = 1 + 2^{-2m}\|w\|^2,$$

and the lemma follows. ∎

A typical vector in $\Lambda_m$ is given by $(u, 2^{-m}(u,w))$ for an integer vector $u$. To help the reader we reserve "$u$" with various indices to be an integer vector of dimension $n+1$ and use $v$ (again with indices) for a typical vector in $\Lambda_m$.

We are interested in running LLL on $\Lambda_m$ for increasing values of $m$. Increasing the value of $m$ does not increase the length of any vector in the lattice and shrinks the length of most vectors. One could hope that LLL finds a strictly better basis as $m$ increases but as it is a complicated and iterative algorithm it is hard to compare the outcomes for different values of $m$. To facilitate the comparison we start the algorithm for parameter $m+1$ on the output found for $m$. On top of this we keep short vectors we have already found intact. Let $(v_i)_{i=1}^{n+1}$ be a basis found for $\Lambda_m$ and let $v_i^*$ be the component of $v_i$ orthogonal to the first $i-1$ vectors in the basis.

**Definition 3.2** *A vector $v_i$ is* short *if $\|v_i^*\| \le 2^{n-i/2}$.*

Note that in an LLL-reduced basis if $v_i$ is short for $i > 1$ then so is $v_{i-1}$. The basic procedure to investigate our weights is now as follows.

- For $m = 0, 1, \ldots \ell$
- Run LLL until it halts but only modifying the non-short vectors $v_i$.
- Update $\Lambda_m$ to $\Lambda_{m+1}$ by dividing the last coordinate of all vectors by 2.

Another way to view the fact that we do not change the short vectors is that once we have found a short vector, $v_i$, we project the remaining vectors to the space orthogonal to $v_i$ and keep running LLL on the smaller space. Once LLL has halted we lift the resulting basis to a basis of the full lattice and make it size reduced.

Let $m_i$ be the value of $m$ when the $i$th short vector appears and let the corresponding vector be $v_i = (u_i, 2^{-m_i}(u_i, w))$. Note that while $u_i$ do not depend on the value of $m$ this is not true for $v_i$ and hence $v_i^*$. To be completely precise we should use the notation $v_i^{*,m}$ but for notational convenience we drop the superscript and leave the value of $m$ implicit.

**Lemma 3.3** *At an increase of $m$ by one, the length of $v_j^*$ does not increase and may not decrease by more than a factor of two.*

**Proof:** The length of $v_j^*$ is the distance from $v_j$ to the linear span of the first $j-1$ basis vectors. As increasing the value of $m$ does not increase the length of any vector and decreases the length of any vector by at most two, the lemma follows. ∎

Consider the integral vectors $u_i$ which are the first $n+1$ coordinates of $v_i$ and define $u_i^*$ as the component of $u_i$ orthogonal to $u_1, u_2, \ldots, u_{i-1}$. Let $\alpha_i$ be $\log_2 |u_i^*|$.

**Lemma 3.4** *For any $j$ we have $\alpha_j \leq n$.*

**Proof:** We have

$$\|u_i^*\| \leq \|u_i\| \leq \|v_i\|,$$

and by size reduction

$$\|v_i\|^2 \leq \frac{1}{4}\sum_{j=1}^{i-1} \|v_j^*\|^2 + \|v_i^*\|^2 \leq 2^{2n}(\frac{1}{4}\sum_{j=1}^{i-1} 2^{-j} + 2^{-i}) \leq 2^{2n}.$$

■

Define

$$t_j = 3nj - \sum_{i=1}^{j-1} \alpha_i.$$

We have a few preliminary observations.

**Lemma 3.5** *For any $s$, $1 \leq s \leq n+1$, we have $0 \leq \sum_{i=1}^{s} \alpha_j \leq sn$.*

**Proof:** Let $U$ be the matrix of size $s \times (n+1)$ with rows $u_i$. Consider $UU^T$. It is a non-singular integral matrix and thus has determinant of absolute value at least 1. As the length of each $u_i$ is at most $2^n$ the determinant is at most $2^{2sn}$. We also have that the determinant is $2^{2\sum_{i=1}^{s} \alpha_i}$ and the lemma follows. ■

**Lemma 3.6** *For any $j$ we have $t_{j+1} \geq 2n + t_j$.*

**Proof:** Immediate from Lemma 3.4 and the definition of $t_j$. ■

**Lemma 3.7** *For $\ell \geq 4n^2 + 4n$ we have $t_{n+1} < m_{n+1}$.*

**Proof:** The determinant of $\Lambda_{m_{n+1}}$ is, by Lemma 3.1 slightly larger than $2^{\ell - m_{n+1}}$. As it has a basis of short vectors this determinant is bounded from above by

$$\prod_{i=1}^{n+1} \|v_i^*\| \leq 2^{n^2+n}$$

and thus $m_{n+1} > \ell - (n^2+n)$. On the other hand, by Lemma 3.5, we know that $t_{n+1} \leq 3n^2 + 3n$, and the lemma follows. ■

Let $k$ be the smallest value such that $m_{k+1} \geq t_{k+1}$. The case when $k = 0$ is slightly degenerate and to avoid discussing this case in the future let us immediately see how to decrease the size of the weights when $k = 0$. Define $w_i' = w_i/2$ where each coordinate is rounded down to an integer. The below lemma finishes the case $k = 0$.

**Lemma 3.8** *If $m_1 > 0$ then $f_w = f_{w'}$.*

**Proof:** For any $x \in \{-1,1\}^{n+1}$ we have $(w',x) = (w,x)/2 + \delta$ where $|\delta| \leq (n+1)/2$. If $|(w,x)| > n+1$ then $f_w(x) = f_{w'}(x)$ and suppose this is not the case. Then $(x,(w,x))$ is a vector of length at most $2n$ in $\Lambda_0$. This implies that LLL applied to $\Lambda_0$ finds a vector of length at most $2n2^{n/2}$ yielding $m_1 = 0$ contradicting the assumption of the lemma. ∎

We turn to the case when $k \geq 1$. Let $W^1$ be the linear span of $(u_i)_{i=1}^k$ and let $W^2$ be its orthogonal complement. We write $w = w^1 + w^2$ where $w^i \in W^i$. To see where we are going let us immediately say that the new set of weights $w'$ is of the form $w^1 + \tilde{w}^2$ where $\tilde{w}^2$ is a vector in $W^2$ close to $w^2/4$. We start by establishing that $w^1$ is fairly short.

**Lemma 3.9** *For $1 \leq i \leq k$ we have $|(u_i^*, w^1)| \leq 2^{t_i+n+1}$.*

**Proof:** We prove this by induction over $i$. For $i=1$ we have $(u_1^*, w^1) = (u_1, w)$ which has absolute value at most $2^{m_1+n} \leq 2^{t_1+n}$.

For the general case write $u_i = u_i^* + \sum_{j=1}^{i-1} c_j u_j^*$ where

$$|c_j| \|u_j^*\| \leq \|u_i\| \leq 2^n,$$

and hence $|c_j| \leq 2^{n-\alpha_j}$. Note that

$$(u_i^*, w^1) = (u_i, w^1) - \sum_{j=1}^{i-1} c_j (u_j^*, w^1)$$

and as, by induction,

$$|c_j(u_j^*, w^1)| \leq |c_j| 2^{t_j+n+1} \leq 2^{t_j+2n+1-\alpha_j} = 2^{t_{j+1}+1-n}$$

we get the upper bound

$$2^{m_i+n} + \sum_{j=1}^{i-1} 2^{t_{j+1}+1-n}.$$

By Lemma 3.6, the sum is at most $2^{t_i+2-n}$ and as $m_i \leq t_i$, the lemma follows. ∎

We have the following corollary.

**Corollary 3.10** *We have $\|w^1\| \leq 2^{t_{k+1}+1-2n}$.*

**Proof:** As $u_i^*$ is a full set of orthogonal vectors in $W^1$ a bound of the form $|(u_i^*, w^1)| \leq A\|u_i^*\|$ valid for all $i$ is sufficient to establish that $\|w^1\| \leq A$. By Lemma 3.9 it is sufficient that $A \geq 2^{t_i+n+1-\alpha_i}$ for all $i$. Finally we note that $t_i - \alpha_i = t_{i+1} - 3n$ and, also using Lemma 3.6, this completes the proof. ∎

As stated above, we would want to have $w' = w^1 + w^2/4$ but this is not an integral vector and we need to find a close vector with integer coordinates. Let $L^2$ be the lattice of points in $W^2$ with integer coordinates. This is a lattice of dimension $n + 1 - k$. We have the following lemma.

**Lemma 3.11** *For any $x \in W^2$ we can efficiently find $y \in L^2$ such that*

$$\|x - y\| \leq n^2 2^{(k+1)n}.$$

**Proof:** We find a full set of independent vectors in $L^2$ and use Lemma 2.1. Let $U$ be the matrix of dimension $k \times (n+1)$ with rows $u_i$. As the $u_i$ are linearly independent this is of rank $k$ and for notational convenience let us assume that the first $k$ columns of $U$ are linearly independent. Let $U^1$ be the square $k \times k$ matrix of these first $k$ columns and let $U^2$ be the other $n + 1 - k$ columns. Consider a vector $x = (x^1, x^2)$ where $x^1$ denotes the first $k$ coordinates. Then $x \in W^2$ iff $x^1 = -U_1^{-1} U_2 x_2$. Let $D$ be the determinant of $U_1$. If $e_i$ is the $i$th unit vector then, by Cramer's rule, for $1 \leq i \leq n + 1 - k$ , the vectors

$$b_i = (-U_1^{-1} U_2 D e_i, D e_i)$$

are linearly independent vectors with integer coordinates and hence lie in $L^2$. Any coordinate of $DU_1^{-1}$ is bounded by $2^{kn}$ and hence any coordinate of $b_i$ is bounded by $n2^{(k+1)n}$. The lemma now follows from Lemma 2.1. ∎

Take the vector $3w^2/4$ and apply Lemma 3.11 to get $\tilde{w}^2 \in L^2$ such that $\|\tilde{w}^2 - 3w^2/4\| \leq n^2 2^{(k+1)n}$. Define $w' = w - \tilde{w}^2$ which is an integral vector close to $w^1 + w^2/4$. We claim that $f_{w'} = f_w$ and proceed to prove that the sign of $(w, x)$ equals the sign of $(w', x)$ for any $x \in \{-1, 1\}^{n+1}$.

Suppose first that $x \in W^1$. Then, as $\tilde{w}^2 \in W^2$, it is true that $(x, w) = (x, w')$ and hence the interesting case is when $x$ is linearly independent of $(u_1)_{i=1}^k$.

**Lemma 3.12** *If $x \in \{-1, 1\}^{n+1}$ does not belong to $W^1$ we have $|(x, w)| \geq 2^{m_{k+1}-1}$.*

**Proof:** If the conclusion of the lemma does not hold then $(x, 2^{1-m_{k+1}}(w, x))$ is a vector of length at most $\sqrt{n+2}$ in $L_{m_{k+1}-1}$ and LLL would have found a vector independent of $v_1, v_2 \ldots v_k$ of length at most $\sqrt{n+2}2^{(n-k)/2}$ and in particular would have discovered the $k + 1$st short vector in $\Lambda_{m_{k+1}-1}$. ∎

By Corollary 3.10 we have

$$|(x, w^1)| \leq \sqrt{n+1} 2^{t_{k+1}+1-2n} \tag{1}$$

and hence

$$|(x, w^2)| \geq 2^{m_{k+1}-1} - \sqrt{n+1} 2^{t_{k+1}+1-2n} \geq 2^{m_{k+1}-2}. \tag{2}$$

We also have

$$|(x, \tilde{w}^2 - 3w^2/4)| \leq (n+1) n^2 2^{(k+1)n} \leq 2^{t_{k+1}-kn}. \tag{3}$$

By (1) and (2) it follows that the sign of $(x, w)$ equals the sign of $(x, w^2)$. Furthermore, as

$$(x, w') = (x, w^1) + (x, w^2/4) + (x, 3w^2/4 - \tilde{w}^2)$$

by (1), (2), and (3), it follow that the sign of $(x, w')$ equals the sign of $(x, w^2)$ and hence also the sign of $(x, w)$ and we conclude that $f_w(x) = f_{w'}(x)$ and hence the two functions are equal for all inputs. Finally let us check that the size of the weights have decreased.

**Lemma 3.13** *If $\|w\| \geq 2^{4n^2+4n}$, then $\|w'\| \leq \|w\|/2$.*

**Proof:** We have that

$$\|w'\| \leq \|w^1\| + \|w^2/4\| + \|3w^2/4 - \tilde{w}^2\| \leq 2^{t_{k+1}+1-2n} + \|w\|/4 + n^2 2^{(k+1)n} \leq \|w\|/2.$$

■

Let us end by stating our main theorem.

**Theorem 3.14** *Suppose we are given a set of weights, $w$, defining a threshold function in $f_w$ in $n$ variables. Then in polynomial time we can find a set of weights $w'$ such that $\|w'\| \leq 2^{4n^2+4n}$ and $f_w = f_{w'}$.*

**Proof:** The idea is to repeatedly use our algorithm preserving the function $f_w$ and decreasing the size of the weights in each iteration by a factor of at least 2. This is a polynomial time algorithm which gives an output as required by the theorem.

The only problem is that the reasoning above requires that $n$ is sufficiently large, i.e. $n > n_0$ for some absolute constant $n_0$. Small values of $n$ are, however, easy to deal with. When $n \leq n_0$ finding the optimal size $w$ is an integer linear program in a constant number of variables. Such a program can be solved in polynomial time [Len83] and hence we can find weights as small as those guaranteed by [Mur71]. As this bound is $(n+1)^{(n+1)/2}2^{1-n}$ which is smaller than the bound claimed in the theorem, the proof is complete. ■

## 4 Final words

We use the lattice reduction algorithm LLL to get simple expressions. Using a stronger lattice reduction that still runs in polynomial time such as the one proposed by Schnorr [Sch87] we would get slightly better bounds but more complicated formulas.

The algorithm as written is not very efficient in that we only decrease the size of $w$ by a factor 2 in each iteration. Changing parameters slightly it is easy to instead get a decrease of $2^{\Omega(n)}$. In a different dimension, lattice reduction in practice gives significantly shorter vectors than what is guaranteed by the theoretical analysis. It is our hope that a carefully tuned variant of the algorithm in this paper would give a very efficient algorithm in practice which finds weights that are considerable smaller than what is stated in our main theorem.

It would be nice to complement our algorithm by a hardness result. We did not consider this problem seriously but let us make some remarks. First we can note that given $w$ and $w'$ it is co-NP complete to determine whether $f_w = f_{w'}$. This follows as given $w \in \mathbb{Z}^n$ using $(1, 2w)$ and $(-1, 2w)$ give the same function unless there is a solution to $\sum_{i=1}^n x_i w_i = 0$ with $x \in \{-1, 1\}^n$.

Finding close to optimal weights intuitively seems like a harder problem than checking whether two sets of weights give the same function and hence one would suspect that this is a provably hard problem. On the other hand, as it is difficult to even verify a good answer it might be hard to prove that finding small and and equivalent weights is NP-hard. A better approximation algorithm for finding the shortest vector in a lattice would almost certainly improve the bounds for the problem studied in this paper. It is not out of the question that there is a connection in the other direction and this could be investigated.

Some LTFs can be defined by weight vectors with very small integers. We do not currently see how to use the existence of such a good representation to find better representations compared to what we can find for general LTFs.

To sum up, there are many open problems in connection with efficiently finding small representations of LTFs and the current paper might only be a first step in a long journey.

**Acknowledgment.** I am grateful to Marc Vinyals for asking the question on how to efficiently find small coefficients to represent LTFs.

## References

[Ajt98] Miklós Ajtai. The shortest vector problem in l2 is np-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19, 1998.

[AV97] Noga Alon and Van H. Vu. Anti-hadamard matrices, coin weighing, threhold gates and indecomposable hypergraphs. *Journal Combinatorial Theory Ser. A*, 79:133–160, 1997.

[Hå94] Johan Håstad. On the size of weights for threshold gates. *SIAM Journal on Discrete Mathematics*, 7:484—492, 1994.

[Kho05] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM (JACM)*, 52(5):789–808, 2005.

[Len83] Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8:538–548, 1983.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Comlexity of Lattice Problems: A Cryptographic Perspective.* Springer Science & Business Media, 2002.

[Mur71] Saburo Muroga. *Threshold Logic and its Applications.* Wiley-Interscience, New York, 1971.

[Sch87] Claus Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53:201–224, 1987.