An Exponential Improvement on the Memorization Capacity of Deep Threshold Networks

Shashank Rajput
University of Wisconsin - Madison
rajput3@wisc.edu

Kartik Sreenivasan
University of Wisconsin - Madison
ksreenivasa2@wisc.edu

Dimitris Papailiopoulos
University of Wisconsin - Madison
dimitris@papail.io

Amin Karbasi
Yale University
amin.karbasi@yale.edu

Abstract

It is well known that modern deep neural networks are powerful enough to memorize datasets even when the labels have been randomized. Recently, Vershynin (2020) settled a long standing question by Baum (1988), proving that *deep threshold* networks can memorize n points in d dimensions using $\widetilde{\mathcal{O}}(e^{1/\delta^2} + \sqrt{n})$ neurons and $\widetilde{\mathcal{O}}(e^{1/\delta^2}(d + \sqrt{n}) + n)$ weights, where δ is the minimum distance between the points. In this work, we improve the dependence on δ from exponential to almost linear, proving that $\widetilde{\mathcal{O}}(\frac{1}{\delta} + \sqrt{n})$ neurons and $\widetilde{\mathcal{O}}(\frac{d}{\delta} + n)$ weights are sufficient. Our construction uses Gaussian random weights only in the first layer, while all the subsequent layers use binary or integer weights. We also prove new lower bounds by connecting memorization in neural networks to the purely geometric problem of separating n points on a sphere using hyperplanes.

1 Introduction

The current paradigm of training neural networks is to train the networks until they fit the training dataset perfectly (Zhang et al., 2017; Arpit et al., 2017). This means that the network is able to output the exact label for every sample in the training set, a phenomenon known as *interpolation*. Quite interestingly, modern deep networks have been known to be powerful enough to interpolate even randomized labels (Zhang et al., 2017; Liu et al., 2020), a phenomenon that is usually referred to as *memorization* (Yun et al., 2019; Vershynin, 2020; Bubeck et al., 2020), where the networks can interpolate any arbitrary labeling of the dataset.

Given that memorization is a common phenomenon in modern deep learning, a reasonable question to ask is that of how big a neural network needs to be so that it can memorize a dataset of n points in d dimensions. This question has been of interest since the 60s (Cover, 1965; Baum, 1988; Mitchison & Durbin, 1989; Sontag, 1990; Huang et al., 1991; Sartori & Antsaklis, 1991). In particular, Baum (1988) proved that a single hidden layer threshold network with $\mathcal{O}(\max(n, d))$ weights and $\mathcal{O}(\lceil n/d \rceil)$ neurons can memorize any set of n points in d dimensions, as long as they are in general position.[1] Baum asked if going deeper could increase the memorization power of threshold networks, and in particular if one could reduce the number of neurons needed for memorization to $\mathcal{O}(\sqrt{n})$. While for ReLU activated networks, Yun et al. (2019) were able to prove that indeed only $\mathcal{O}(\sqrt{n})$ neurons were sufficient for memorization using deeper networks, the same question remained open for threshold networks.

[1] The "general position" assumption of n vectors in $\mathbb{R}^d$ means that any collection of d of these n vectors are linearly independent.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Reference	**# Neurons**	**# Weights**	**Assumptions**
Baum (1988)	$\mathcal{O}(\lceil n/d \rceil)$	$\mathcal{O}(\max(n,d))$	General Position
Huang et al. (1991)	$\mathcal{O}(n)$	$\mathcal{O}(nd)$	None
Vershynin (2020)	$\mathcal{O}(e^{1/\delta^2}\log^p(n)+\sqrt{n}\log^{2.5}(n))$	$\mathcal{O}(e^{1/\delta^2}(d+\sqrt{n})+n\log^5(n))$	δ-separated, points lie on the unit sphere
Ours, Theorem 1	$\mathcal{O}\left(\frac{\log^2 n}{\delta}+\sqrt{n}\log^2 n\right)$	$\mathcal{O}\left(\frac{(d+\log n)\log n}{\delta}+n\log^2 n\right)$	δ-separated

Table 1: Table comparing the upper bounds for the parameters needed for a threshold network to memorize a dataset. General position assumption in $\mathbb{R}^d$ means that no more than d points lie on a $d-1$ dimensional hyperplane. Note: $\log^p(n)$ denotes a poly-log factor in n.

Recently, Vershynin (2020) was able to answer Baum's question in the positive by proving that indeed $\widetilde{\mathcal{O}}(e^{1/\delta^2}+\sqrt{n})$ neurons were sufficient to memorize a set of n points on a unit sphere separated by a distance at least δ. Many recent works study the memorization power of neural networks under separation assumptions, *e.g.,* Bubeck et al. (2020); Vershynin (2020); Park et al. (2020). One reason why before Vershynin's work, it was unclear whether going deeper was helpful for threshold networks was that unlike ReLU or sigmoid functions, the threshold activation function, *i.e.,* $\sigma(z)=\mathbf{1}_{z\geq 0}$, prohibits neurons from passing 'amplitude information' to the next layer.

Although the dependence of $O(\sqrt{n})$ on the number of neurons was achieved by the work of Vershynin, it is unclear that the exponential dependence on distance and the requirement that the points lie on a sphere is fundamental. In this work, we lift the spherical requirement and offer an exponential improvement on the dependence on δ.

Theorem 1. *(Informal) There exists a threshold neural network with $\widetilde{\mathcal{O}}\left(\frac{1}{\delta}+\sqrt{n}\right)$ neurons and $\widetilde{\mathcal{O}}\left(\frac{d}{\delta}+n\right)$ weights that can memorize any δ-separated dataset of size n in d dimensions.*

Please see Definition 2 for the formal definition of δ-separation; informally, it means that all the points have bounded norm and a distance of at least δ between them. Comparing the theorem above with Vershynin's result, we see that firstly we have reduced the dependence on δ from exponential to nearly linear; and secondly, in our upper bound for the number of weights, the δ and n terms appear in summation rather than product (ignoring the logarithmic factors). Further, note that Vershynin (2020) needs the points to lie on a sphere, whereas we only need them to have a bounded ℓ^2 norm. We have compared our results with the existing work in Table 1.

Our construction has $\mathcal{O}(\log\frac{\log n}{\delta})$ layers, of which only the first layer in our construction has real weights, which are i.i.d Gaussian. This layer has $\mathcal{O}(\frac{\log n}{\delta})$ neurons and $\mathcal{O}(\frac{d\log n}{\delta})$ weights. All the other layers have integer weights which are bounded and are on the of order $\widetilde{\mathcal{O}}(\log n)$.

Baum (1988) also proved that there exists a dataset of n points such that any threshold neural network would need at least $\lceil n/d \rceil$ neurons in the first layer to memorize it. However, the distance for this dataset is $\mathcal{O}(1/n)$.[2] Later, Baum & Haussler (1989) proved VC-dimension bounds which imply that $\min\{\sqrt{n}, n/d\}$ neurons are necessary in a network to memorize a dataset of n points in d dimensions, and this bound is independent of the minimum distance δ. Our upper bound (Theorem 1) shows that if $\delta=\Omega(1/n)$, then we can memorize with only $\mathcal{O}(\frac{\log n}{\delta})$ neurons in the first layer. To complement our upper bound, we introduce a new, δ-dependent lower bound below:

Theorem 2. *(Informal) There exists a δ-separated dataset of size $n\in\left[\frac{d^2}{\delta},\left(\frac{1}{\delta}\right)^{\frac{d}{2}}\right]$ such that any threshold network that can memorize it needs $\widetilde{\Omega}\left(\frac{1}{\sqrt{\delta}}\right)$ neurons in the first layer.*

The rest of the paper is divided into 7 sections. In Section 2, we provide the related works for memorization using neural networks. We provide definitions and notation in Section 3 and our main results in Section 4. Then, we briefly explain the constructions of our upper bound in Section 5.

[2]Here, we have rescaled the dataset to have maximum norm of any sample to be 1. This is done to make the dataset consistent with our minimum distance definition (see Assumption 2). The assumption in Park et al. (2020) also uses this kind of normalization.

Before exploring lower bounds for threshold networks, we provide sharp bounds on the minimum parameterization needed for any model (not necessarily a neural network) to memorize a δ-separated dataset, in Section 6. We discuss our lower bound for threshold networks in Section 7. Finally, we conclude our paper with a brief conclusion in Section 8.

2 Related works

Memorization with threshold activation. There is a rich history of research into the memorization capacity of threshold networks. Baum (1988) showed that a single hidden layer network with $\mathcal{O}(\max(n, d))$ weights and $\mathcal{O}(\lceil n/d \rceil)$ neurons are sufficient to memorize a dataset, *if* the samples are in general positions. Baum also provided an instance of a dataset, where any threshold based network aiming to memorize the dataset would need at least $\lceil n/d \rceil$ neurons in the first layer. However, the construction was such that δ scaled inversely with n. In this work, we show that for any minimum distance $0 < \delta \leq \frac{1}{2}$, the number of neurons in the first layer can be much smaller. Baum also proved, using the Function Counting Theorem (Cover, 1965), that $\mathcal{O}(n/\log n)$ connections are needed in any neural network that aims to memorize points in general position, regardless of the minimum distance.

Huang et al. (1991) and Sartori & Antsaklis (1991) proved that threshold networks could memorize any arbitrary set of points, that is, the points need not be in general position. However, these results needed $(n-1)d$ weights and $n-1$ neurons in the networks for memorization. Mitchison & Durbin (1989) and Kowalczyk (1997) provided upper and lower bounds when the requirement of memorizing the dataset exactly was relaxed to allow a recall error of up 50%. Sontag (1990) provided evidence that suggested that adding skip connections can double the memorization capacity of single hidden layer networks. Kotsovsky et al. (2020) extended Baum's result to multiclass classification setting, using bithresholding activation functions.

Recently, Vershynin (2020) was able to show that $\mathcal{O}(e^{1/\delta^2}(d+\sqrt{n})+n)$ weights and $\mathcal{O}(e^{1/\delta^2}+\sqrt{n})$ neurons are sufficient to memorize a set of n points on the unit sphere, separated by a distance of at least δ. This answered the question by Baum (1988), which was whether the number of neurons could be reduced to $\mathcal{O}(\sqrt{n})$ by considering deeper networks. Note that Vershynin's and Baum's assumptions are fundamentally different and neither is stronger than the other. For example, Baum's construction would still need only $\mathcal{O}(\max(n, d))$ weights even if the points are infinitesimally close to each other, whereas Vershynin's construction would grow as δ decreases. On the other hand, Vershynin's construction works even if all the points lie on a low dimensional hyperplane, whereas Baum's general position assumption would get violated in this situation. Note that naïvely extending Baum's construction to the case where the points lie on a very low dimensional hyperplane would result in $\widetilde{\mathcal{O}}(nd)$ weights and $\widetilde{\mathcal{O}}(n)$ neurons[3], which is identical to the bounds by Huang et al. (1991).

Memorization with ReLU and sigmoid activation. Due to the popularity of the ReLU activation function, there has also been an interest in their memorization capacity. Zhang et al. (2017) proved that a single hidden layer ReLU network with $2n+d$ weights and n neurons can memorize n arbitrary points. Hardt & Ma (2017) proved that ReLU networks with residual connections also need only $\mathcal{O}(n)$ weights to memorize n points. Yun et al. (2019) were able to prove that for ReLU, in fact, one needs only $\mathcal{O}(\sqrt{n})$ neurons to memorize n points. Quite interestingly, the construction by Vershynin (2020) works for both threshold activated networks (as described above) as well as ReLU activated networks; and gives the same bounds for both architectures. Bubeck et al. (2020) show that under some minimum distance assumptions, ReLU networks can memorize (even real labels) with $\mathcal{O}(n/d)$ neurons such that the total weight (Bartlett, 1998) of the network is $\mathcal{O}(\sqrt{n})$, which is proved to be optimal for single hidden layer networks. Huang (2003) proved that a two-hidden layer network with $\mathcal{O}(\sqrt{n})$ sigmoid activated neurons suffices to memorize n points. Recently, Park et al. (2020) have shown that ReLU or sigmoid activated networks with only $\mathcal{O}(n^{2/3}+\log(1/\delta))$ weights are sufficient to memorize n points separated by a normalized distance of δ.

Over parameterization and generalization. A large body of recent work attempts to explain why large neural networks, which are capable memorizers, can also generalize well, *e.g.,* Neyshabur et al. (2018, 2019); Belkin et al. (2020). In this work, we only study memorization, and omit any discussion about generalization. We do not believe our results explain anything about generalization.

[3]Please see Appendix A for a detailed explanation.

3 Preliminaries

In this work, we will consider feed forward neural networks $f : \mathbb{R}^d \to \{0, 1\}$ of the form

$$f(\boldsymbol{x}) := \sigma(b_L + \boldsymbol{w}_L^\top \sigma(\boldsymbol{b}_{L-1} + \boldsymbol{W}_{L-1}\sigma(\boldsymbol{b}_{L-2} + \boldsymbol{W}_{L-2}\sigma(\dots\sigma(\boldsymbol{b}_1 + \boldsymbol{W}_1\boldsymbol{x})\dots)))),$$

where $\sigma(\cdot)$ is the threshold activation. The network has L layers, with the last layer consisting of a single neuron with bias $b_L \in \mathbb{R}$, and weight vector $\boldsymbol{w}_L$. For any other layer l, its bias vector is $\boldsymbol{b}_l$ and weight matrix is $\boldsymbol{W}_l$. We use the notation $[n]$ to denote the set $\{1, \dots, n\}$.

We formally define memorization as follows:

Definition 1. *A learning algorithm $\mathcal{A}$ can memorize a dataset of feature vectors $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$ if for any arbitrary labeling $\{y_i\}_{i=1}^n$ of the dataset, the algorithm can output a model f such that $\forall i \in [n] : f(\boldsymbol{x}_i) = y_i$.*

For instance, $\mathcal{A}$ could be the training procedure for a particular neural architecture and f could be the neural network that $\mathcal{A}$ outputs after the training process. For neural networks, we will abuse the notation and say that a neural network f can memorize a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$ if for every arbitrary labeling $\{y_i\}_{i=1}^n$ of the dataset, we can find a set of weights and biases for the network so that $\forall i \in [n] : f(\boldsymbol{x}_i) = y_i$.

We are mainly interested in the minimum size of a threshold network so that it can memorize a dataset of n samples. Our only assumption on the dataset will be that of δ-separation, which we define below.

Definition 2. *We say that a dataset is δ-separated if it satisfies either Assumption 1 **or** Assumption 2.*

Assumption 1. *(Angular separation) All the feature vectors in the dataset, $\{\boldsymbol{x}_i\}_{i=1}^n$ satisfy*

$$\arccos\left(\frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_j\rangle}{\|\boldsymbol{x}_i\|\|\boldsymbol{x}_j\|}\right) \geq \delta$$

Assumption 2. *(Normalized minimum distance) All the feature vectors in the dataset, $\{\boldsymbol{x}_i\}_{i=1}^n$ satisfy $\|\boldsymbol{x}_i\| \leq 1$ and*

$$\forall i \neq j : \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \delta.$$

Remark 1. *Note that the 'normalized' in the name of Assumption 2 refers to the fact that the maximum norm of any point in the dataset is bounded by 1. Our results can be easily extended to any other bounded dataset by rescaling the weights and biases of the first layer appropriately. Park et al. (2020) also use a similar normalized distance assumption.*

Note that if the points lie on the unit sphere, like in Vershynin (2020) or Theorem 2 in this paper, then both assumptions are roughly equivalent.

Notation: We use the lower case letters for scalars (w), lower case bold letters for vectors ($\boldsymbol{w}$), and upper case bold letters for matrices ($\boldsymbol{W}$). We denote the i-th element of vector $\boldsymbol{w}$ as w_i, similarly for the case when the vectors have subscript, we denote the i-th element of $\boldsymbol{w}_j$ by $w_{j,i}$. We use a tilde over $\mathcal{O}(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$, *i.e.,* $\widetilde{\mathcal{O}}(\cdot)$, $\widetilde{\Omega}(\cdot)$ and $\widetilde{\Theta}(\cdot)$, to hide the logarithmic factors.

4 Main results

As discussed earlier, the existing result by Vershynin (2020) has an exponential dependence on δ. In this section, we provide a new and tighter upper bound on the number of weights and neurons needed for a memorization, which brings the dependence down to almost linear. Our result is stated below:

Theorem 1. *There exists a threshold activated neural network with $\mathcal{O}(\frac{\log^2 n}{\delta} + \sqrt{n}\log^2 n)$ neurons and $\mathcal{O}\left(\frac{(d+\log n)\log n}{\delta} + n\log^2 n\right)$ weights that can memorize any dataset of size n in d dimensions that is δ-separated.*

In this construction, only the first layer has real weights. The rest of the network has binary and integer weights. In particular, the $O(n\log^2 n)$ term in the number of weights comes from binary and integer weights (with the integers being bounded and on the order of $\mathcal{O}(\log n)$). Hence, the $\mathcal{O}(n\log^2 n)$

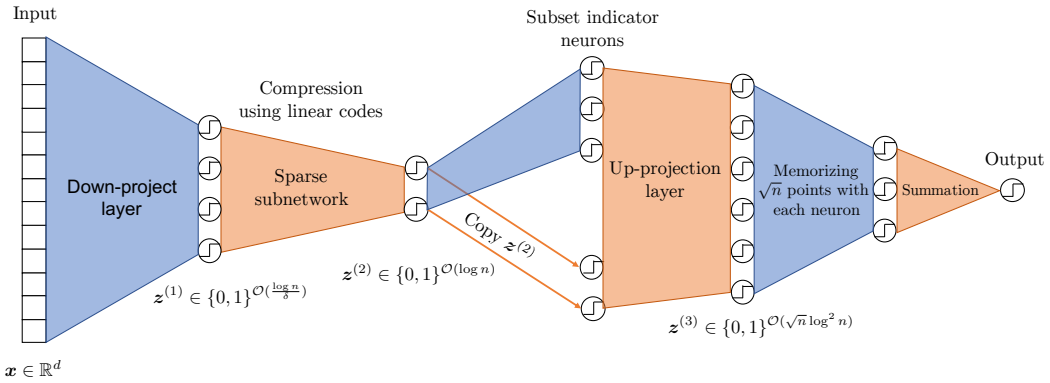

Figure 1: The schematic of the neural network that can memorize any δ-separated dataset of size n in d dimensions (Theorem 1). The first layer has random Gaussian weights and projects the input from $\mathbb{R}^d$ down to $\{0,1\}^{\mathcal{O}(\frac{\log n}{\delta})}$. The next component is a sparse subnetwork that further compresses the vectors from $\{0,1\}^{\mathcal{O}(\frac{\log n}{\delta})}$ to $\{0,1\}^{\mathcal{O}(\log n)}$ using linear codes. The next couple of layers lift the vectors from $\{0,1\}^{\mathcal{O}(\log n)}$ to $\{0,1\}^{\mathcal{O}(\sqrt{n}\log^2 n)}$ while ensuring that we can find sets of vectors of size $\sqrt{n}$ that are linearly independent. Finally, we use this linear independence to memorize up to $\sqrt{n}\log^2 n$ points with a single neuron. This gives us a layer of size $\sqrt{n}$ that can memorize the entire dataset. The final layer is just a single neuron that sums the outputs of these $\sqrt{n}$ neurons.

weights can actually be represented by $\mathcal{O}(n\log^2 n \log\log n)$ bits. We will see in Section 6 that $\Omega(n)$ bits are necessary for memorization by any model and hence this term is optimal, up to logarithmic factors. The $\frac{d\log n}{\delta}$ weights in the bound are the weights of the first layer which has width $\frac{\log n}{\delta}$, for which later, in Theorem 2, we will provide some indication that this factor might also be necessary.

Baum (1988) had constructed a dataset with normalized minimum distance $\mathcal{O}(1/n)$ and proved that for this dataset, any threshold network aiming to memorize this must have $\lceil n/d \rceil$ neurons. We introduce a new, δ-dependent lower bound below:

Theorem 2. *For every $0 < \delta \leq \frac{1}{2}$ and some universal constants C_1 and C_2, there exists a dataset of size $n \in \left[\frac{C_1 d^2 \log^2(d/\delta)}{\delta}, \left(\frac{C_2}{\delta}\right)^{\frac{d}{2}}\right]$ satisfying the δ-separation condition such that any threshold network that can memorize it needs $\Omega\left(\frac{1}{\sqrt{\delta}\log(1/\delta)}\right)$ neurons in the first layer.*

Note that the dataset in this theorem actually satisfies *both* Assumption 1 and Assumption 2, up to a constant factor. To understand the implication of Theorem 2 a bit better, consider two dataset $\mathcal{D}_1$ and $\mathcal{D}_2$, both containing n points on the sphere. We assume that $\mathcal{D}_1$ has δ_1-separation and $\mathcal{D}_2$ has δ_2-separation, where $\delta_1 = \Theta(n^{-2/d})$ and $\delta_2 = \widetilde{\Theta}(d^2/n)$. Then, Theorem 1 says that there exists a network that can memorize $\mathcal{D}_1$ with only $\mathcal{O}(n^{2/d}\log n)$ neurons in the first layer, whereas Theorem 2 says that any network aiming to memorize $\mathcal{D}_2$ would need at least $\widetilde{\Omega}(\sqrt{n}/d)$ neurons in the first layer. Hence, we see that the minimum distance can have a big impact on the network architecture.

Comparing Theorems 1 and 2, we see that there is a gap in the bounds: Theorem 1 proves an upper bound of $\widetilde{\mathcal{O}}(1/\delta)$ neurons, whereas Theorem 2 proves a lower bound of $\widetilde{\Omega}(1/\sqrt{\delta})$ neurons. We think that the gap is an artifact of the proof of Theorem 2, and a tighter analysis or a better construction could result in a lower bound of $\widetilde{\Omega}(1/\delta)$ neurons, which will match the upper bound of Theorem 1. However, this is just speculation based on intuition, and more research will be needed to answer this open question.

5 Memorization with threshold networks

In this section, we give a proof sketch of Theorem 1, the complete proof is provided in the appendix. We will construct a network that can memorize any δ-separated dataset. As we discussed before,

the threshold activation prohibits the passing of amplitude information to deeper layers. This is the biggest obstacle in leveraging the benefits of multiple layers of transformation that can be obtained by networks with activations such as ReLU. In the following, we will explain our construction layer-wise:

Step 1: Generating unique binary representations. Given that the amplitude information is lost after thresholding, we want to ensure that at the least, the first layer is able to transform the inputs into binary representations such that each sample in our dataset has a unique binary representation. Later, we will see that this is sufficient for memorization.

Now we give an overview of how a layer with $\mathcal{O}(\frac{\log n}{\delta})$ neurons with random i.i.d. weights can convert the input into unique binary vectors of length $\mathcal{O}(\frac{\log n}{\delta})$. In this sketch, we will do this under Assumption 1, but the same technique can be extended for Assumption 2.

The key result we use is that any hyperplane passing through the origin, with random Gaussian coefficients has at least a $\delta/2\pi$ probability of separating two points which have an angle of at least δ between them. To see how this is true, consider two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ with an angle δ between them. Consider the 2-dimensional space spanned by these two vectors. Note that the intersection of the Gaussian hyperplane with the 2-dimensional space spanned by $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is just a line passing through the origin. Further, because the Gaussian hyperplane has isometric coefficients, it can be shown that the line has angle uniformly distributed in $[0, 2\pi)$. This implies that the probability that this line passes in between the two points is at least $\delta/2\pi$. If we take m such independent Gaussian hyperplanes, then the probability that none of them separate a pair $\boldsymbol{x}_i, \boldsymbol{x}_j$ is less than $(1 - \frac{\delta}{2\pi})^m$. Finally, taking a simple union bound over the $\binom{n}{2}$ pair of points, shows that the probability that at least one pair has no hyperplane passing in between them is at most

$$\binom{n}{2}\left(1 - \frac{\delta}{2\pi}\right)^m.$$

We want this to be less than 1 to show that there exists one set of hyperplanes which can separate all the pairs. Doing that gives us $m = \mathcal{O}(\frac{\log n}{\delta})$, which is our required bound.

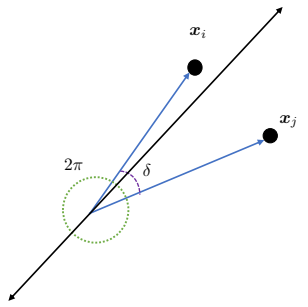

Figure 2: The probability that a uniformly random line passing through the origin passes in between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is $\delta/2\pi$.

For any input $\boldsymbol{x}_i$, each threshold neuron in the first layer outputs either 0 or 1. We can concatenate the outputs of all the neurons in the first layer into a binary vector. Let us denote this binary representation of $\boldsymbol{x}_i$ created by the first layer as $\boldsymbol{z}_i^{(1)}$, for $i = 1, \ldots, n$. Further, denote the dimension of $\boldsymbol{z}_i^{(1)}$ by $d^{(1)}$, which is equal to the number neurons in the first layer, that is, $d^{(1)} = \mathcal{O}(\frac{\log n}{\delta})$.

Step 2: Compressing the binary representations to optimal length. In order to avoid a factor δ in the later layers, our next task is to further compress the $\mathcal{O}(\frac{\log n}{\delta})$ dimensional vectors returned by the first layer into vectors of length $\mathcal{O}(\log n)$. Note that $\mathcal{O}(\log n)$ length is optimal (up to constant multiplicative factors) if we want to represent the n samples uniquely using binary vectors. For this we will use linear codes where we simulate XOR operations by threshold activations.

Consider two $d^{(1)}$-dimensional binary vectors $\boldsymbol{z}_i^{(1)}$ and $\boldsymbol{z}_j^{(1)}$ that differ in at least one bit. Let $\oplus$ represent the XOR operation. Let $\boldsymbol{b}$ be a random binary vector of length $d^{(1)}$ where each bit is i.i.d Bernoulli(0.5). Then, we claim that

$$\Pr(\langle \boldsymbol{z}_i^{(1)}, \boldsymbol{b}\rangle_{\oplus} \neq \langle \boldsymbol{z}_j^{(1)}, \boldsymbol{b}\rangle_{\oplus}) = 0.5,$$
$$\text{where } \langle \boldsymbol{z}, \boldsymbol{b}\rangle_{\oplus} = (z_1 \cdot b_1) \oplus \cdots \oplus (z_{d^{(1)}} \cdot b_{d^{(1)}})$$

For the ease of exposition, assume that $\boldsymbol{z}_i^{(1)}$ and $\boldsymbol{z}_j^{(1)}$ differ in only the first bit (the proof of the general case is provided in the appendix). Let the first bit of $\boldsymbol{z}_i^{(1)}$ be 0, and that of $\boldsymbol{z}_j^{(1)}$ be 1, and we have assumed that the rest of their bits are the same. Let $c := (z_{i,2}^{(1)} \cdot b_2) \oplus \cdots \oplus (z_{i,d^{(1)}}^{(1)} \cdot b_{d^{(1)}}) =$

$(z_{j,2} \cdot b_2) \oplus \cdots \oplus (z_{j,d^{(1)}} \cdot b_{d^{(1)}})$. Then,

$$\Pr(\langle \boldsymbol{z}_i^{(1)}, \boldsymbol{b} \rangle_\oplus \neq \langle \boldsymbol{z}_j^{(1)}, \boldsymbol{b} \rangle_\oplus) = \Pr(((0 \cdot b_1) \oplus c) \neq ((1 \cdot b_1) \oplus c)).$$

It can now be verified that if $b_1 = 1$, then $((0 \cdot b_1) \oplus c) \neq ((1 \cdot b_1) \oplus c)$. Since $\Pr(b_1 = 1) = 0.5$, we get the desired result.

We have shown that the probability that one such random vector $\boldsymbol{b}$ differentiates between a pair $\boldsymbol{z}_i^{(1)}$ and $\boldsymbol{z}_j^{(1)}$ with probability 0.5. Doing an analysis similar to the one that we did for the first layer, we get that $m = \mathcal{O}(\log n)$ vectors suffice to separate all pairs. The big task here, however, is to implement the XOR operation using threshold activations. In the appendix, we show that this can in fact be achieved using a couple of sparse layers.

Similar to the output of the first layer, let us denote the compressed binary representation of $\boldsymbol{z}_i^{(1)}$ created by these sparse layers as $\boldsymbol{z}_i^{(2)}$, for $i = 1, \ldots, n$. Further, let $\boldsymbol{z}_i^{(2)}$ have dimension $d^{(2)}$, where $d^{(2)} = \mathcal{O}(\log n)$, as we showed above.

Step 3: Partitioning the samples into subsets of size $\mathcal{O}(\sqrt{n} \log n)$. Once we have $d^{(2)} = \mathcal{O}(\log n)$ length binary representations of the inputs, memorizing with a single layer consisting of $(d^{(2)} + 1)n$ weights and n neurons is not difficult. However, our aim is to reduce the number of neurons to $\mathcal{O}(\sqrt{n} \log n)$. For that, we use a strategy similar to Yun et al. (2019): we will memorize up to K samples with 1 neuron, where $K = \Omega(\sqrt{n} \log n)$. Thus, we will need only $\mathcal{O}(\sqrt{n})$ neurons in total. Roughly speaking the strategy would be the following: Let's say we have a neuron $\sigma(\boldsymbol{w}^\top \boldsymbol{z} + b)$. Then, if for any $\{y_1, \ldots, y_K\} \in \{0, 1\}^K$, we can find a $\boldsymbol{w}$ and b such that all of the following equations hold at the same time,

$$\boldsymbol{w}^\top \boldsymbol{z}_1 + b = y_1, \quad \ldots \quad , \boldsymbol{w}^\top \boldsymbol{z}_K + b = y_K, \quad (1)$$

then this neuron can memorize the K samples $\{\boldsymbol{z}_1, \ldots \boldsymbol{z}_K\}$. To do so, we need that the vectors $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K\}$ be linearly independent[4]. Since we have $K = \Omega(\sqrt{n} \log n)$, we require that $\boldsymbol{z}_i$'s have dimension at least K. However, the previous few layers have compressed the samples into $d^{(2)} = \mathcal{O}(\log n)$ length binary vectors. Thus, the next task is to project these $d^{(2)}$ length binary vectors to $\Omega(K)$ length binary vectors. Note that first compressing down to $\mathcal{O}(\log n)$ length vectors and then expanding these up to $\Omega(\sqrt{n} \log n)$ length binary vectors seems counter intuitive. However, we were unable to project the vectors directly to $\Omega(\sqrt{n} \log n)$ length binary vectors without incurring a large dependence on δ and n in the number of weights and neurons.

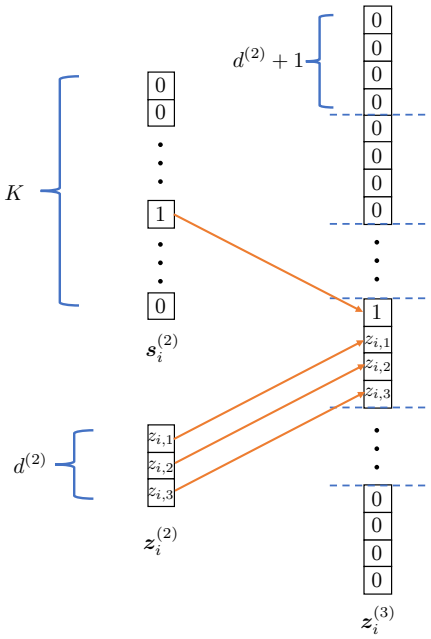

Figure 3: In **Step 3**, $\boldsymbol{z}_i^{(2)}$ is copied to a location in $\boldsymbol{z}_i^{(3)}$ depending on the subset S_j to which $\boldsymbol{z}_i^{(2)}$ belongs.

One more objective that we accomplish while projecting up to $\Omega(K)$ dimension is that both the projection and memorizing the resulting vectors can be easily done with bounded integer weights. The way we do this is by partitioning the n vectors $\boldsymbol{z}_i^{(2)}$ into K subsets of size at most $\sqrt{n}$ each. We prove that we can find such K subsets, such that each is characterized by a unique prefix, which all the elements of that subset share. These prefixes ensure that we can easily detect which subset a particular $\boldsymbol{z}_i^{(2)}$ belongs to, using threshold neurons. For more details on this, please refer to the appendix.

[4]Yun et al. (2019) also use this strategy. However, as we will see in Step 4, we are able to ensure that the vectors $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K\}$ are linearly independent while also ensuring that their coordinates do not overlap. This helps us find a vector $\boldsymbol{w}$ which has integer values and satisfies (1). Further, since Yun et al. (2019) use ReLU activations, the way they achieve linear independence, and the construction of the rest of the network is different.

Let $\{S_1, \ldots, S_K\}$ denote the K subsets, each with a unique prefix. Let $\boldsymbol{z}_i^{(2)} \in \{0,1\}^{d^{(2)}}$ be the binary vector returned by **Step 2** above for input $\boldsymbol{x}_i$. We mentioned above that by using threshold neurons, we can create a vector $\boldsymbol{s}_i^{(2)} \in \{0,1\}^K$ for $\boldsymbol{z}_i^{(2)}$ such that $s_{i,j} = 1$ if and only if $\boldsymbol{z}_i^{(2)} \in S_j$. Then, using these prefixes and threshold neurons, we can create the expanded binary vector $\boldsymbol{z}_i^{(3)} \in \{0,1\}^{K(d^{(2)}+1)}$ corresponding to the vector $\boldsymbol{z}_i^{(2)}$. The way we create $\boldsymbol{z}_i^{(3)}$ from $\boldsymbol{z}_i^{(2)}$ is as follows: $\boldsymbol{z}_i^{(3)}$ is partitioned into K chunks of length $d^{(2)}+1$. If $\boldsymbol{z}_i^{(2)} \in S_j$, then the first bit of the j-th chunk is set to 1 and $\boldsymbol{z}_i^{(2)}$ is copied to the rest of the bits in that chunk. All the other bits in $\boldsymbol{z}_i^{(3)}$ are 0. This is shown in Figure 3. This operation can be done with threshold activated neuron, for details please refer to the appendix.

Step 4: Memorizing the samples with $\sqrt{n}$ neurons A key property about the vectors $\boldsymbol{z}_i^{(3)}$ is that if we take K such vectors, each belonging to a different subset S_j, then they will be linearly independent as their coordinates do not overlap. Thus, we can find $\boldsymbol{w}$ and b such that (1) is satisfied. Furthermore, due to the fact that these K vectors have different supports, we can manage to satisfy (1) with $\boldsymbol{w}$ which have integer elements on the order of $\mathcal{O}(\log n)$.

Hence, we can memorize up to K samples with one neuron. This gives that we need $\sqrt{n}$ in the second-to-last layer to memorize all the samples. The final layer is simply a single neuron that sums the outputs of the second-to-last layer.

Note that a threshold neuron can be approximated using three ReLU neurons. Hence, the construction described in this section can also be used to create a ReLU activated network with can memorize the same dataset.

5.1 Extending the construction to multiple class memorization

The construction above can be easily extended to the case when there are more than two classes. Let there be $M \geq 2$ classes. Then, the label for each sample can be represented as a sequence of $\log_2 M$ bits. In this case, we can replicate the construction of the last and the second-to-last layers $\log_2 M$ times so that the new network has $\log_2 M$ output neurons, each output neuron memorizing one bit of the label. Hence, we get the following Corollary:

Corollary 1. *There exists a threshold activated neural network with $\mathcal{O}(\frac{\log^2 n}{\delta} + \sqrt{n}(\log^2 n)(\log M))$ neurons and $\mathcal{O}\left(\frac{(d+\log n)\log n}{\delta} + n(\log^2 n)(\log M)\right)$ weights that can memorize any M-class dataset of size n in d dimensions that is δ-separated.*

6 Bit complexity of memorization

Before understanding the minimum size that threshold networks have to be in order to memorize a dataset, a more fundamental question is how many bits are needed to specify the models output by any learning algorithm that aims to memorize a dataset. In this section, we provide two theorems that sharply characterize the bit complexity needed for memorization. The results provide the bounds in the terms of two very related quantities, the *packing number* and the *covering number*, which we define below.

Definition 3. *Given a real number $\delta > 0$, the packing number $\mathcal{P}_\delta$ of a set $\mathcal{S}$ (equipped with norm $\|\cdot\|$) is defined as the cardinality of the largest set $\mathcal{X} \subset \mathcal{S}$ such that*

$$\forall x_1, x_2 \in \mathcal{X} : \|x_1 - x_2\| \geq \delta.$$

Intuitively, the packing number is the maximum number of $\delta/2$ radius non-overlapping balls that one can fit inside a set.

Definition 4. *Given a real number $\delta > 0$, the covering number $\mathcal{C}_\delta$ of a set $\mathcal{S}$ (equipped with norm $\|\cdot\|$) is defined as the cardinality of the smallest set $\mathcal{X} \subset \mathcal{S}$ such that*

$$\forall x \in \mathcal{S}, \exists \hat{x} \in \mathcal{X} : \|x - \hat{x}\| \leq \delta.$$

Intuitively, the covering number is the minimum number of δ radius (possibly overlapping) balls that one needs to completely cover a set.

These quantities are very useful in quantifying the 'size' of a set. These are related to each other through the following relation (Vershynin, 2018, Lemma 4.2.8):

$$\mathcal{P}_{2\delta} \leq \mathcal{C}_\delta \leq \mathcal{P}_\delta.$$

Now, we are ready to state the upper and lower bounds on the bit complexity of memorization.

Theorem 3. *Let $\mathcal{S}$ be a set from which we select the samples. Let $\mathcal{A}$ be a learning algorithm that can memorize any dataset of feature vectors $\mathcal{D} \in \mathcal{S}^n$ as long as every pair of feature vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}$ satisfies $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \delta$. Then, we need at least $\max(n, \log_2 \log_2 \mathcal{P}_\delta)$ bits to represent the models output by $\mathcal{A}$.*

Theorem 4. *Let $\mathcal{S}$ be a set from which we select the samples. Then, there exists a learning algorithm $\mathcal{A}$ that can memorize any dataset of feature vectors $\mathcal{D} \in \mathcal{S}^n$ as long as every pair of feature vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}$ satisfies $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \delta$, and the models returned by $\mathcal{A}$ can be represented in $O(n + \log_2 \log_2 \mathcal{C}_{\delta/2})$ bits.*

Note that since $a + b = \mathcal{O}(\max(a, b))$ for non-negative variables a and b, the upper bound from Theorem 4 is of the same order as the lower bound from Theorem 3.

It is known that the packing and covering numbers of the unit sphere are of the order $\Theta((C_\text{p}/\delta)^d)$ and $\Theta((C_\text{c}/\delta)^d)$, for some universal constants C_p and C_c respectively. Substituting this into Theorems 3 and 4, we get the following corollaries.

Corollary 2. *Let $\mathcal{S}^{d-1}$ be the unit sphere in $\mathbb{R}^d$. Let $\mathcal{A}$ be a learning algorithm that can memorize any set of n points on $\mathcal{S}^{d-1}$ as long as every pair of points $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}$ satisfies $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \delta$. Then, we need $\Omega(\max(n, \log d + \log\log \frac{1}{\delta}))$ bits to represent the models output by $\mathcal{A}$.*

Corollary 3. *Let $\mathcal{S}^{d-1}$ be the unit sphere in $\mathbb{R}^d$. Then, there exists a learning algorithm $\mathcal{A}$ that can memorize any set of n points on $\mathcal{S}^{d-1}$ as long as every pair of feature vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}$ satisfies $\|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \delta$, and the models returned by $\mathcal{A}$ can be represented in $O(n + \log d + \log\log \frac{1}{\delta})$ bits.*

Note that if we have a set of n points on the unit sphere separated by at least distance δ, then the set satisfies *both* Assumption 1 and Assumption 2. Hence, these two corollaries give bounds on the bit complexity of memorization under both the δ-separation assumptions.

7 Lower bound construction

For our lower bound construction, we will have n points on the unit sphere in d-dimensions. Each neuron in the first layer represents a hyperplane in d-dimensions. The main observation that we use is that for every pair of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ such that they have different labels, there should be a hyperplane passing in between them. Thus, we aim to lower bound the minimum number of hyperplanes needed for this to happen. This is related to the problem of having to separate every point using hyperplanes, that is, the problem where we need to ensure that there is a hyperplane passing in between *every* pair of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. For this problem we have the following theorem.

Theorem 5. *For every $0 < \delta \leq \frac{1}{2}$ and some universal constants C_1 and C_2, there exists a set of n points on the d-dimensional sphere, with $n \in \left[\frac{C_1 d^2 \log^5(d/\delta)}{\delta}, \left(\frac{C_2}{\delta}\right)^{\frac{d}{2}}\right]$, such that each pair of points is separated by a distance of at least δ, and one needs $\Omega\left(\frac{\log n}{\sqrt{\delta}\log\frac{\log n}{\delta}}\right)$ hyperplanes to separate them.*

We use essentially the same construction for both the problems. On the sphere, first we sample $\sqrt{n}$ points uniformly at random, which we call *cluster centers*. Around

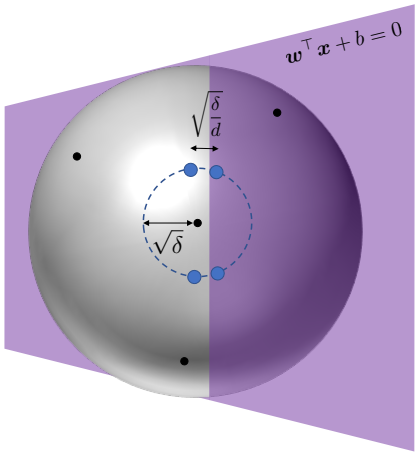

Figure 4: In this figure, the black dots represent the cluster centers and the blue circles represent the samples in a cluster. In high dimensions, the points in each cluster get concentrated within distance $\mathcal{O}(\sqrt{\delta/d})$ in any direction. Hence any hyperplane, that wants to separate a constant fraction of the points in a cluster, needs to pass within distance $\mathcal{O}(\sqrt{\delta/d})$ of the cluster center, as shown in the figure.

each of these centers, we sample $\sqrt{n}$ points uniformly at random from the cap of radius $\sqrt{\delta}$, and call these sets of points *clusters*. Hence, in total we sample n points on the sphere and it can be shown that with high probability these will all be at least δ distance away from each other as long as $n \in \left[\frac{d^2}{\delta}, \frac{1}{\delta^{d/2}}\right]$. In high dimensions, it can be shown that for each cluster and in any direction, most points lie within a distance $\sqrt{\delta/d}$ of the cluster center. Hence, if a hyperplane aims to 'effectively' separate pairs of points in a cluster, it needs to pass within distance $\sqrt{\delta/d}$ of the cluster center. It can also be shown that in high dimensions any hyperplane can only pass within distance $\sqrt{\delta/d}$ of the cluster centers of $\sqrt{\delta}$ fraction of the clusters. Finally, we use the following necessary inequality to get a bound on the required number of hyperplanes:

$$\begin{aligned}&(\#\text{ hyperplanes needed})\\ &\geq \frac{(\#\text{ clusters}) \times (\#\text{ hyperplanes needed per cluster})}{(\#\text{ clusters a hyperplane can 'effectively' separate})}.\end{aligned}$$

We know that the number of clusters is $\sqrt{n}$, and we showed above that $(\#$ clusters a hyperplane can 'effectively' separate$)$ is $\sqrt{\delta}$ fraction of the clusters, that is $\mathcal{O}(\sqrt{n\delta})$. Finally, for the term $(\#$ hyperplanes needed per cluster$)$, we use 1 for the memorization problem, where we need to separate points with opposite label. For the problem of separating every pair of points, we know that $(\#$ hyperplanes needed per cluster$) = \Omega(\log_2 \sqrt{n})$ since there $\sqrt{n}$ points in each cluster and hence we need at least $\log_2 \sqrt{n}$ hyperplanes per cluster. Substituting these values gives us the bounds in Theorems 2 and 5.

Note that we skipped many details in the sketch above. The complete proof is provided in the appendix, and the construction there is slightly different.

8 Conclusion

In this work, we study the memorization capacity of threshold networks under the δ-separation assumption. We improve the existing bounds on the number of neurons and weights required, from exponential in δ, to almost linear. We also prove new, δ-dependent lower bounds for the memorization capacity of threshold networks, that together with our upper bound, shows that δ (the separation) indeed impacts the network architecture required for memorization.

Acknowledgments and Disclosure of Funding

DP acknowledges the support of NSF CAREER Award #1844951, Sony Faculty Innovation Award, AFOSR & AFRL Center of Excellence Award FA9550-18-1-0166, NSF TRIPODS Award #1740707, and ONR YIA Grant N00014-21-1-2806. AK acknowledges the support of NSF CAREER Award IIS-1845032, and ONR Grant N00014-19-1-240.

References

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pp. 233–242. PMLR, 2017.

Richard Arratia and Louis Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of mathematical biology*, 51(1):125–131, 1989.

Keith Ball et al. An elementary introduction to modern convex geometry. *Flavors of geometry*, 31: 1–58, 1997.

Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536, 1998.

Eric B Baum. On the capabilities of multilayer perceptrons. *Journal of complexity*, 4(3):193–215, 1988.

Eric B Baum and David Haussler. What size net gives valid generalization? *Neural computation*, 1 (1):151–160, 1989.

Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, 2(4):1167–1180, 2020.

Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, and Dan Mikulincer. Network size and weights size for memorization with two-layers neural networks. *arXiv preprint arXiv:2006.02855*, 2020.

Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, 1965.

Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. Essential coding theory. *Draft available at http://www. cse. buffalo. edu/ atri/courses/coding-theory/book*, 2012.

Moritz Hardt and Tengyu Ma. Identity matters in deep learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Guang-Bin Huang. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE transactions on neural networks*, 14(2):274–281, 2003.

Shih-Chi Huang, Yih-Fang Huang, et al. Bounds on the number of hidden neurons in multilayer perceptrons. *IEEE transactions on neural networks*, 2(1):47–55, 1991.

Vladyslav Kotsovsky, Fedir Geche, and Anatoliy Batyuk. Bithreshold neural network classifier. In *2020 IEEE 15th International Conference on Computer Sciences and Information Technologies (CSIT)*, volume 1, pp. 32–35. IEEE, 2020.

Adam Kowalczyk. Estimates of storage capacity of multilayer perceptron with threshold logic hidden units. *Neural networks*, 10(8):1417–1433, 1997.

Shengchao Liu, Dimitris Papailiopoulos, and Dimitris Achlioptas. Bad global minima exist and sgd can reach them. *Advances in Neural Information Processing Systems*, 33, 2020.

GJ Mitchison and RM Durbin. Bounds on the learning capacity of some multi-layer networks. *Biological Cybernetics*, 1989.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2018.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Sejun Park, Jaeho Lee, Chulhee Yun, and Jinwoo Shin. Provable memorization via deep neural networks using sub-linear parameters. *arXiv preprint arXiv:2010.13363*, 2020.

Michael A Sartori and Panos J Antsaklis. A simple method to derive bounds on the size and to train multilayer neural networks. *IEEE transactions on neural networks*, 2(4):467–471, 1991.

Eduardo D Sontag. Remarks on interpolation and recognition using neural nets. In *NIPS*, pp. 939–945, 1990.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Roman Vershynin. Memory capacity of neural networks with threshold and rectified linear unit activations. *SIAM Journal on Mathematics of Data Science*, 2(4):1004–1033, 2020.

Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small relu networks are powerful memorizers: a tight analysis of memorization capacity. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 15532–15543, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Please refer to the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Comparing Vershynin (2020) and Baum (1988)

In this section, we show how the assumptions in Vershynin (2020) and Baum (1988) are not comparable, and in different situations one result can be better than the other.

First we show an example where the points are well-separated, but do not lie in general position. Let $d = \Omega(\log n)$. Note that n points can be arranged in a $\mathcal{O}(\log n)$-dimensional unit sphere while still ensuring that the distance between them is a constant, that is, δ is a universal constant. Hence, even though the ambient space is d-dimensional, the dataset can lie on a $\mathcal{O}(\log n)$-dimensional sphere, and be separated by a constant distance. To visualize this, consider how in a 3-dimensional space, a dataset could lie on a circle (which is 2-dimensional). In this setting, Vershynin's bound would still need $\widetilde{\mathcal{O}}(\sqrt{n})$ neurons and $\widetilde{\mathcal{O}}(n+d)$ weights. On the other hand, naively extending Baum's construction to this setting would require $\mathcal{O}(n/\log n)$ neurons and thus $\mathcal{O}(nd/\log n)$ weights.

Next, consider the case when the points lie in general position, but the distance between them is infinitesimally small. In this case, Vershynin's bounds containing the e^{1/δ^2} term will be much larger than Baum's bounds which are independent of minimum distance.

Finally, if the points are indeed well separated *and* in general position, then Vershynin's construction will be better when $d \ll \sqrt{n}$, and Baum's construction will be better when $d \gg \sqrt{n}$.

Note that the comparison between our results and Baum's results would be identical to the comparison done above.

B Proof of Theorem 1

Proof. As explained in Section 5, the network memorizes in four steps. In this proof, we will go into the details of each step.

Step 1: Generating unique binary representations. The first layer's task is to generate unique binary representations for each sample in the dataset. What this means is that for every pair of samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, there should exist at least one neuron in the first layer, say $\sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)$ such that $\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \neq \sigma(\boldsymbol{w}^\top \boldsymbol{x}_j + b)$. Note that $\boldsymbol{w}^\top \boldsymbol{x} + b$ is just a hyperplane and $\sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \neq \sigma(\boldsymbol{w}^\top \boldsymbol{x}_j + b)$ is equivalent to saying that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ lie on the opposite sides of the hyperplane. The next two lemmas says that we can easily find a set of $\mathcal{O}(\log(n)/\delta)$ hyperplanes such that for every pair of points $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, at least one hyperplanes passes in between them.

Lemma 1. *Assume that our dataset satisfies Assumption 1. Define* $m := \left\lceil \frac{4\pi}{\delta} \log\left(\frac{n}{\epsilon}\right) \right\rceil$ *for any* $\epsilon \in (0,1)$. *If we sample* m *hyperplanes of the form* $\mathbf{w}_i^\top \boldsymbol{x} = 0$, *for* $i = 1, \ldots, m$, *where* $\mathbf{w}_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, *then with probability* $1 - \epsilon$,

$$\forall i \neq j, \exists k : (\mathbf{w}_k^\top \boldsymbol{x}_i)(\mathbf{w}_k^\top \boldsymbol{x}_j) < 0.$$

Lemma 2. *Assume that our dataset satisfies Assumption 2. Define* $m := \frac{C}{\delta} \log\left(\frac{n}{\epsilon}\right)$ *for some universal constant* C *and any* $\epsilon \in (0,1)$. *If we sample* m *hyperplanes of the form* $\mathbf{w}_i^\top \boldsymbol{x} + \mathrm{b}_i = 0$, *for* $i = 1, \ldots, m$, *where* $\mathbf{w}_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$ *and* $\mathrm{b}_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$, *then with probability* $1 - \epsilon$,

$$\forall i \neq j, \exists k : (\mathbf{w}_k^\top \boldsymbol{x}_i + \mathrm{b}_k)(\mathbf{w}_k^\top \boldsymbol{x}_j + \mathrm{b}_k) < 0.$$

Note that each neuron in the first layer outputs either 0 or 1. If we concatenate all the outputs of these $\mathcal{O}(\log(n)/\delta)$ neurons, we will get a binary vector of length $\mathcal{O}(\log(n)/\delta)$. The lemmas above say that under Assumption 1 *or* Assumption 2, these binary vectors will be unique.

Let $\boldsymbol{z}_i^{(1)}$ be the binary vector created by the first layer when the input to the network is $\boldsymbol{x}_i$. Further, let $d^{(1)}$ denote the dimension (length) of this binary vector. Note that the first layer has $\mathcal{O}(\log(n)/\delta)$ neurons and $\mathcal{O}(d\log(n)/\delta)$ weights and biases.

Step 2: Compressing the binary representations. The next few layers compress the $\mathcal{O}(\log(n)/\delta)$ length binary vectors down to $\mathcal{O}(\log(n))$ length unique binary representations for each sample. We will use linear codes in $\mathbb{F}_2$ (the finite field of two elements: 0 and 1) for this.

For two binary vectors $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \{0,1\}^{d^{(1)}}$, define the inner product in the field as

$$\langle \boldsymbol{b}_1, \boldsymbol{b}_2 \rangle_{\oplus} := (b_{1,1} \cdot b_{2,1}) \oplus (b_{1,2} \cdot b_{2,2}) \oplus \cdots \oplus (b_{1,d^{(1)}} \cdot b_{2,d^{(1)}}),$$

where $\oplus$ is the XOR operator, and $b_{1,i}$ is the i-th element of $\boldsymbol{b}_1$, and similarly $b_{2,i}$ is the i-th element of $\boldsymbol{b}_2$.

Let $\mathbf{b}_1, \ldots, \mathbf{b}_m$ be m i.i.d. Bernoulli(0.5) vectors of length $d^{(1)}$ each. Define vectors $\boldsymbol{z}_i^{(2)}$ as

$$\boldsymbol{z}_i^{(2)} = \begin{bmatrix} \langle \mathbf{b}_1, \boldsymbol{z}_i^{(1)} \rangle_{\oplus} \\ \vdots \\ \langle \mathbf{b}_m, \boldsymbol{z}_i^{(1)} \rangle_{\oplus} \end{bmatrix}, \forall i \in [n]. \tag{2}$$

The next lemma says that we only need $m = \mathcal{O}(\log n)$ to ensure that $\forall i \neq j$, $\boldsymbol{z}_i^{(2)} \neq \boldsymbol{z}_j^{(2)}$.

Lemma 3. *Let $\boldsymbol{z}_1^{(1)}, \ldots, \boldsymbol{z}_n^{(1)}$ be n distinct binary vectors of dimension $d^{(1)}$, and let $\mathbf{b}_1, \ldots, \mathbf{b}_m$ be m vectors in $d^{(1)}$ dimensions, with i.i.d. Bernoulli(0.5) entries. Let $\boldsymbol{z}_1^{(2)}, \ldots, \boldsymbol{z}_n^{(2)}$ be as defined in* (2)*. If $m = 3 \log \frac{n}{\epsilon'}$, then with probability $1 - \epsilon'$,*

$$\forall i \neq j : \boldsymbol{z}_i^{(2)} \neq \boldsymbol{z}_j^{(2)}.$$

The lemma above is a special case of the Gilbert-Varshamov bound (for example, see (Guruswami et al., 2012)). A simple proof of the lemma is provided in subsection B.1.3 for completeness.

Hence, we will be able to compress the $d^{(1)}$-dimensional vectors down to m-dimensions. In particular this means we will be able to compress down from $\mathcal{O}(\frac{\log n}{\delta})$ to $\mathcal{O}(\log n)$ dimensions. *However*, note that the key in the operation above is the XOR operator, so we will need to use threshold activation to simulate the XOR operation. The next lemma says that we can transform $\boldsymbol{z}_i^{(1)}$ to $\boldsymbol{z}_i^{(2)}$ using a threshold network containing $\mathcal{O}(\frac{\log^2 n}{\delta})$ neurons and $\mathcal{O}(\frac{\log^2 n}{\delta})$ weights.

Lemma 4. *Let $\boldsymbol{z}_i^{(2)}$ be as defined in* (2)*, for all $i \in [n]$. Then, we can construct a threshold network with $3d^{(1)}m$ neurons and $9d^{(1)}m$ weights and biases such that it outputs $\boldsymbol{z}_i^{(1)}$ when the input to it is $\boldsymbol{z}_i^{(2)}$, for all $i \in [n]$.*

Hence, when the input to the overall network is $\boldsymbol{x}_i$, the network constructed up till now will output $\boldsymbol{z}_i^{(2)}$. Similar to the first layer, let $d^{(2)} = m$ denote the dimension of $\boldsymbol{z}_i^{(2)}$. Note that this step uses $\mathcal{O}(\frac{\log^2 n}{\delta})$ neurons and $\mathcal{O}(\frac{\log^2 n}{\delta})$ weights.

Step 3: Partitioning the dataset into $\widetilde{\mathcal{O}}(\sqrt{n})$ subsets of size at most $\sqrt{n}$ each. In this step, we partition $\{\boldsymbol{z}_1^{(2)}, \ldots, \boldsymbol{z}_n^{(2)}\}$ into $\mathcal{O}(\sqrt{n} \log n)$ subsets of size at most $\sqrt{n}$ each. The reason why we do this will be apparent in **Step 4**. The key property that we want these subsets to have is that each subset should have a unique prefix such that all the $\boldsymbol{z}_i^{(2)}$ that belong to one subset share that prefix. This will help in detecting which subset a particular $\boldsymbol{z}_i^{(2)}$ belongs to, using threshold networks. Hence, first we will show that such prefixes exist.

Lemma 5. *Given n distinct binary vectors $\boldsymbol{z}_1^{(2)}, \ldots, \boldsymbol{z}_n^{(2)}$ of length $d^{(2)} = c \log_2 n$ each (for some constant $c \geq 1$), we can partition them into $\mathcal{O}(\sqrt{n} \log n)$ subsets of size at most $\sqrt{n}$ each, with the important property that each of these subsets will be characterized by unique prefixes, such that all of the elements of one subset will have the same prefix.*

Lemma 5 gives us $K = \mathcal{O}(\sqrt{n} \log n)$ subsets which partition $\{\boldsymbol{z}_1^{(2)}, \ldots, \boldsymbol{z}_n^{(2)}\}$, each with a unique prefix. Let the subsets be $S_1, \ldots, S_K$. Let the binary prefix for S_i be $\boldsymbol{b}_i$, which a length $1 \leq l \leq d^{(2)}$. Then the membership of a vector $\boldsymbol{z}$ can easily be detected by the neuron $s_i : \{0,1\}^{d^{(2)}} \to \{0,1\}$ given by $s_i(\boldsymbol{z}) = \sigma(\sum_{j=1}^{l} (2b_{i,j} - 1) z_j - t)$, where t is the number of 1's in $\boldsymbol{b}_i$. Hence, we can have K neurons, one for each subset, such that $s_i(\boldsymbol{z}) = 1$ if and only if $\boldsymbol{z} \in S_i$. Let $\boldsymbol{s}(\boldsymbol{z})$ represent the

vector formed after concatenating the outputs of all these neurons. Along with these K neurons, we will also have $d^{(2)}$ other neurons that simply copy $\boldsymbol{z}^{(2)}$ over to the next layer. These $d^{(2)}$ neurons can simply have the form $f_i(\boldsymbol{z}) = \sigma(2z_i - 1)$, for $i = 1, \ldots, d^{(2)}$. Overall, if $\boldsymbol{z}_i^{(2)} \in S_j$, then this layer outputs

$$\begin{bmatrix} \boldsymbol{s}(\boldsymbol{z}_i^{(2)}) \\ \boldsymbol{z}_i^{(2)} \end{bmatrix}, \tag{3}$$

as shown in the middle section of Figure 1. Thus, for **Step 3**, we have one layer which consists of K subset indicator neurons, along with $d^{(2)}$ neurons that simply copy $\boldsymbol{z}$ over to the output. Overall, in this step we used one layer with $K + d^{(2)}$ neuron and $(K + 1)d^{(2)}$ weights.

Step 4: The reason we divided the vectors into subsets and created the indicator neurons is so that the next layer can project the $d^{(2)} = \mathcal{O}(\log n)$ dimensional vectors $\boldsymbol{z}_i^{(2)}$ to $d^{(3)} = \mathcal{O}(\sqrt{n}\log^2 n)$ dimension vectors using the subset indicators. Let $\boldsymbol{z}_i^{(3)}$ be the projection of $\boldsymbol{z}_i^{(2)}$. The way we project will ensure that if we take one vector from each of the K subsets, then these vectors would be linearly independent. This linear independence of K vectors would help in memorizing K samples with just one neuron in the second-to-last layer, which means that we can memorize all the n samples with $\sqrt{n}$ neurons in the second-to-last layer. We explain all of this in detail now.

Step 4(a): Projecting the vectors up to dimension $\mathcal{O}(\sqrt{n}\log^2 n)$. In this layer, we will have $K(d^{(2)} + 1)$ neurons partitioned into groups of size $d^{(2)} + 1$ each. As explained in the main paper, the goal would be carry out the transformation shown in Figure 3.

To see how this happens, recall that the previous layer outputs (3). Then, the j-th neuron of the i-th group in this layer would have the form $g_{i,j} = \sigma(s_i - 1)$ for $j = 1$ and $g_{i,j} = \sigma_{i,j}(s_i + z_j^{(2)} - 2)$, for $j = 2, \ldots, d^{(2)} + 1$. It can be verified that this implements the transformation shown in Figure 3.

This layer will have $K(d^{(2)} + 1)$ neurons and $\leq 3K(d^{(2)} + 1)$ weights and biases. Further, looking at neurons, note that the weights and bias are integers and less than 2.

Step 4(b): Memorizing using $\sqrt{n}$ neurons. In this layer, we have $\sqrt{n}$ neurons. The i-th neuron memorizes the i-th samples from each of the K subsets. Let $\boldsymbol{z}_{i,j}^{(2)}$ denote the i-th vector in S_j. The i-th neuron will have the weight vector given by

$$\boldsymbol{w}_i^{(3)} = \begin{bmatrix} -t_{i,1} + y_{i,1} - 1 \\ 2\boldsymbol{z}_{i,1}^{(2)} - \mathbf{1} \\ -t_{i,2} + y_{i,2} - 1 \\ 2\boldsymbol{z}_{i,2}^{(2)} - \mathbf{1} \\ \vdots \\ -t_{i,d^{(2)}} + y_{i,d^{(2)}} - 1 \\ 2\boldsymbol{z}_{i,d^{(2)}}^{(2)} - \mathbf{1} \end{bmatrix},$$

where $t_{i,j}$ is the number of 1's in $\boldsymbol{z}_{i,j}^{(2)}$, and $y_{i,j}$ is the label of sample $\boldsymbol{x}_{i,j}$ from the dataset that corresponds to $\boldsymbol{z}_{i,j}^{(2)}$. The biases of these neurons are 0.

To see how this works, consider the vector $\boldsymbol{z}_{i,j}^{(2)}$. This is the i-th vector in S_j. Then, the corresponding $\boldsymbol{z}_{i,j}^{(3)}$ would have support only in the j-th chunk of length $d^{(2)} + 1$. Then,

$$\begin{aligned} (\boldsymbol{w}_i^{(3)})^\top \boldsymbol{z}_{i,j}^{(3)} &= -t_{i,j} + y_{i,j} - 1 + (2\boldsymbol{z}_{i,j}^{(2)} - \mathbf{1})^\top \boldsymbol{z}_{i,j}^{(2)} \\ &= -t_{i,j} + y_{i,j} - 1 + 2t_{i,j} - t_{i,j} \\ &= y_{i,j} - 1. \end{aligned}$$

Hence, $\sigma((\boldsymbol{w}_i^{(3)})^\top \boldsymbol{z}_{i,j}^{(3)}) = y_{i,j}$. Further, for any $k \neq i$,

$$(\boldsymbol{w}_i^{(3)})^\top \boldsymbol{z}_{k,j}^{(3)} = -t_{i,j} + y_{i,j} - 1 + (2\boldsymbol{z}_{i,j}^{(2)} - \mathbf{1})^\top \boldsymbol{z}_{k,j}^{(2)}.$$

Note that $(2\boldsymbol{z}_{i,j}^{(2)} - \mathbf{1})^\top \boldsymbol{z}_{j,j}^{(2)} \leq t_{i,j} - 1$ if $\boldsymbol{z}_{i,j}^{(2)} \neq \boldsymbol{z}_{k,j}^{(2)}$. Hence,

$$(\boldsymbol{w}_i^{(3)})^\top \boldsymbol{z}_{k,j}^{(3)} \leq y_{i,j} - 2.$$

Hence, $\sigma((\boldsymbol{w}_i^{(3)})^\top \boldsymbol{z}_{k,j}^{(3)}) = 0$.

Thus we see that in this layer, the i-th neuron memorizes the i-th samples from each subset $\{S_1, \ldots, S_K\}$, and for any other sample, it outputs 0. Since there are at most $\sqrt{n}$ samples in each subset, we need $\sqrt{n}$ neurons in this layer.

Let $[q_1, \ldots, q_{\sqrt{n}}]$ be the concatenated output of this layer. Then, the last layer is a single neuron, $\sigma(2q_1 + \cdots + 2q_{\sqrt{n}} - 1)$, which essentially just sums up the outputs of the previous layer. Since each neuron of the previous layer memorized a subset of the samples, this neuron is able to output the correct label for the entire dataset.

□

B.1 Proof of helper lemmas for Theorem 1

B.1.1 Proof of Lemma 1

Proof. Consider any two samples from the dataset, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be two orthonormal vectors that form a basis of the 2-dimensional space spanned by $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Then, any point in the subspace can be written as $z_1\boldsymbol{u} + z_2\boldsymbol{v}$, for some scalars z_1 and z_2. Let $\boldsymbol{x}_i = z_{i,1}\boldsymbol{u} + z_{i,2}\boldsymbol{v}$ and $\boldsymbol{x}_j = z_{j,1}\boldsymbol{u} + z_{j,2}\boldsymbol{v}$.

As per our construction of the hyperplanes, let $\mathbf{w}^\top \boldsymbol{x} = 0$ be any one of the m hyperplanes. Then, we want to find the probability that $(\mathbf{w}^\top \boldsymbol{x}_i)(\mathbf{w}^\top \boldsymbol{x}_j) < 0$. For this, it would be sufficient that $\mathbf{w}^\top \boldsymbol{x}_i > 0$ and $\mathbf{w}^\top \boldsymbol{x}_j < 0$. These are equivalent to $z_{i,1}\mathbf{w}^\top \boldsymbol{u} + z_{i,2}\mathbf{w}^\top \boldsymbol{v} > 0$ and $z_{j,1}\mathbf{w}^\top \boldsymbol{u} + z_{j,2}\mathbf{w}^\top \boldsymbol{v} < 0$. Note that since $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, we have that $\mathbf{w}^\top \boldsymbol{u} \sim \mathcal{N}(0, 1)$ and $\mathbf{w}^\top \boldsymbol{v} \sim \mathcal{N}(0, 1)$, and are independent. For convenience, define the random variables $\mathrm{g}_1 := \mathbf{w}^\top \boldsymbol{u}$, $\mathrm{g}_2 := \mathbf{w}^\top \boldsymbol{v}$, and $\mathbf{g} := [\mathrm{g}_1 \quad \mathrm{g}_2]^\top$. Also, define $\boldsymbol{z}_i = [z_{i,1} \quad z_{i,2}]^\top$ and $\boldsymbol{z}_j = [z_{j,1} \quad z_{j,2}]^\top$. Then, we want to bound the probability of event

$$\{\mathbf{g}^\top \boldsymbol{z}_i > 0 \text{ and } \mathbf{g}^\top \boldsymbol{z}_j < 0\}. \tag{4}$$

Note that $\mathbf{g}$, $\boldsymbol{z}_i$, and $\boldsymbol{z}_j$ are all in 2 dimensions, the angle between $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ is at least δ. Thus, for (4) to be true, $\mathbf{g}$ should lie in a cone of angle δ (see Figure 2). Noting that $\mathbf{g}$ is rotationally symmetric, we get that the probability of (4) is exactly $\frac{\delta}{2\pi}$. Hence, the probability that the hyperplane does not separate $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, is less than $1 - \frac{\delta}{2\pi}$. Since we have m independently sampled hyperplanes, the probability that none of them separates $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ is less than $(1 - \frac{\delta}{2\pi})^m$. This is the probability that one of the pairs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ does not get separated. Since there are a total of $\binom{n}{2}$ pairs of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, then using the union bound, the probability that there exists one pair of points such that the m hyperplanes do not separate them, is less than $\binom{n}{2}(1 - \frac{\delta}{2\pi})^m$. We want this probability to be less than ϵ, that is,

$$\binom{n}{2}\left(1 - \frac{\delta}{2\pi}\right)^m \leq \epsilon.$$

Note that $\binom{n}{2} \leq n^2$ and $\left(1 - \frac{\delta}{2\pi}\right)^m \leq e^{-\frac{\delta}{2\pi}m}$. Hence, it is sufficient for the following to hold

$$n^2 e^{-\frac{\delta}{2\pi}m} \leq \epsilon.$$

This gives that $m \geq \frac{2\pi}{\delta}\log\left(\frac{n^2}{\epsilon}\right)$ is sufficient for the lemma to be true. □

B.1.2 Proof of Lemma 2

This proof is similar to the proof of Lemma 1.

Proof. Consider any two samples from the dataset, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be two orthonormal vectors that form a basis of the 2-dimensional space spanned by $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, such that $\boldsymbol{v} \perp (\boldsymbol{x}_i - \boldsymbol{x}_j)$.

If $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ do not span a 2-dimensional space, then take $\boldsymbol{u}$ to be in the direction of $\boldsymbol{x}_i$, and $\boldsymbol{v}$ can be in any other orthogonal direction. Then, any point in the subspace can be written as $z_1\boldsymbol{u} + z_2\boldsymbol{v}$, for some scalars z_1 and z_2. Let $\boldsymbol{x}_i = z_{i,1}\boldsymbol{u} + z_{i,2}\boldsymbol{v}$ and $\boldsymbol{x}_j = z_{j,1}\boldsymbol{u} + z_{j,2}\boldsymbol{v}$. Note that because $\boldsymbol{v} \perp (\boldsymbol{x}_i - \boldsymbol{x}_j)$, we have that $z_{i,2} = z_{j,2}$, and also $|z_{i,1} - z_{j,1}| = \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \delta$.

As per our construction of the hyperplanes, let $\mathbf{w}^\top \boldsymbol{x} + \mathrm{b} = 0$ be any one of the m hyperplanes. We are interested in the probability that $(\mathbf{w}^\top \boldsymbol{x}_i + \mathrm{b}_i)(\mathbf{w}^\top \boldsymbol{x}_j + \mathrm{b}_j) < 0$, or equivalently

$$(z_{i,1}\mathbf{w}^\top \boldsymbol{u} + z_{i,2}\mathbf{w}^\top \boldsymbol{v} + \mathrm{b})(z_{j,1}\mathbf{w}^\top \boldsymbol{u} + z_{j,2}\mathbf{w}^\top \boldsymbol{v} + \mathrm{b}) < 0. \tag{5}$$

Note that since $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, we have that $\mathbf{w}^\top \boldsymbol{u} \sim \mathcal{N}(0,1)$ and $\mathbf{w}^\top \boldsymbol{v} \sim \mathcal{N}(0,1)$, and are independent. For convenience, define the random variables $\mathrm{g}_1 := \mathbf{w}^\top \boldsymbol{u}$, and $\mathrm{g}_2 := \mathbf{w}^\top \boldsymbol{v}$. Let (s,t) be the coordinates in the 2 dimensional space spanned by $\boldsymbol{u}$ and $\boldsymbol{v}$. Then, note from (5) that we are interested in the behaviour of the hyperplane

$$\mathrm{g}_1 s + \mathrm{g}_2 t + \mathrm{b} = 0, \tag{6}$$

where $\mathrm{g}_1, \mathrm{g}_2$ are the coefficients and b is the bias. Note that this is just a line in 2-dimensions. In particular, (5) is still equivalent to the event that this line passes in between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in this 2-dimensional space.

Looking at the line (6), its slope is $-\mathrm{g}_1/\mathrm{g}_2$ and its (signed) distance from the origin is $\mathrm{b}/\sqrt{\mathrm{g}_1^2 + \mathrm{g}_2^2}$. Next, note that if we consider the two dimensional Gaussian vector $[\mathrm{g}_2 \ \mathrm{g}_1]^\top \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_2)$, then thinking in terms of polar coordinates, $\sqrt{\mathrm{g}_1^2 + \mathrm{g}_2^2}$ is its norm and $-\mathrm{g}_1/\mathrm{g}_2$ is its direction. By the spherical symmetry of Gaussians, we know that the norm and its direction are independent. This implies that $\mathrm{b}/\sqrt{\mathrm{g}_1^2 + \mathrm{g}_2^2}$ is independent of $-\mathrm{g}_1/\mathrm{g}_2$, that is the distance of the line from the origin is independent of its slope. Note that $r := \sqrt{2}\mathrm{b}/\sqrt{\mathrm{g}_1^2 + \mathrm{g}_2^2}$ is the (scaled) ratio of a Gaussian random variable with the square-root of a χ^2-random variable, and hence r has a Student's t-distribution with 2 degrees of freedom.

Fix the slope $\theta := -\mathrm{g}_1/\mathrm{g}_2$ of the line (6), then we have seen that the signed distance from the origin is the independent random variable $\mathrm{r}/\sqrt{2}$. Then, for the line (6) to pass in between $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, we will need to have that $\frac{\mathrm{r}}{\sqrt{2}}$ should be between $\left(\mathrm{k}\frac{z_{i,2}+\theta z_{i,1}}{\sqrt{1+\theta^2}}\right)$ and $\left(\mathrm{k}\frac{z_{j,2}+\theta z_{j,1}}{\sqrt{1+\theta^2}}\right)$, where $\mathrm{k} = -1$ or $\mathrm{k} = 1$ depending on the sign of θ. These two exact values do not matter much. What is more important is that since $\|\boldsymbol{x}_i\| \leq 1$ and $\|\boldsymbol{x}_j\| \leq 1$, we know that these values are less than 1; and that

$$\begin{aligned}\left|\left(\mathrm{k}\frac{z_{i,2}+\theta z_{i,1}}{\sqrt{1+\theta^2}}\right) - \left(\mathrm{k}\frac{z_{j,2}+\theta z_{j,1}}{\sqrt{1+\theta^2}}\right)\right| &= \frac{|\theta|}{\sqrt{1+\theta^2}}|z_{j,1} - z_{i,1}| \\ &\qquad \text{(Since } \mathrm{k} \in \{-1,1\} \text{ and } z_{i,2} = z_{j,2}.\text{)} \\ &\geq \delta\frac{|\theta|}{\sqrt{1+\theta^2}}.\end{aligned}$$

In short, once we fix the slope, θ, the probability that the line passes in between the two points is the same as that of a scaled Student's t-variable's value being in an interval of length at least $\delta\frac{|\theta|}{\sqrt{1+\theta^2}}$. Note that the interval lies completely in $[-1,1]$. Because in the interval $[-\sqrt{2},\sqrt{2}]$, a Student's t-variable has its p.d.f. lower bounded by a univeral constant C, we get that the probability that the line passes in between the two points is at least $2C\sqrt{2}\delta\frac{|\theta|}{\sqrt{1+\theta^2}}$.

To compute a lower bound on the final probability, we need to unfix the slope θ and integrate the lower bound we computed above, over the distribution of the slope. Define the polar angle $\phi := \arctan(\theta) = \arctan(-g_1/g_2)$. Note that due to the spherical symmetry of the Gaussian vector $[\mathrm{g}_1 \ \mathrm{g}_2]$, we will have that ϕ is uniformly distributed in $[0, 2\pi]$. Hence, the final probability is lower bounded by

$$\frac{1}{2\pi}\int_0^{2\pi} C\sqrt{2}\delta\frac{|\theta|}{\sqrt{1+\theta^2}}\mathrm{d}\phi = \frac{1}{2\pi}C\sqrt{2}\delta\int_0^{\pi}|\sin(\phi)|\mathrm{d}\phi = C'\delta,$$

for some universal constant C'.

What we have shown above, is that the probability that a hyperplane separates $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is at least $C'\delta$. Hence, the probability that the hyperplane does not separate $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, is less than $1 - C'\delta$. Since we have m independently sampled hyperplanes, the probability that none of them separate $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, is less than $(1 - C'\delta)^m$. This is the probability that one of the pairs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ does not get separated. Since there are a total of $\binom{n}{2}$ pairs of points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, then by using the union bound, the probability that there exists one pair of points such that the m hyperplanes does not separate them, is less than $\binom{n}{2}(1 - C'\delta)^m$. We want this probability to be less than ϵ, that is,

$$\binom{n}{2}(1 - C'\delta)^m \leq \epsilon.$$

Note that $\binom{n}{2} \leq n^2$ and $(1 - C'\delta)^m \leq e^{-C'\delta m}$. Hence, it is sufficient for the following to hold

$$n^2 e^{-C'\delta m} \leq \epsilon.$$

This gives that $m \geq \frac{1}{C'\delta} \log\left(\frac{n^2}{\epsilon}\right)$ is sufficient for the lemma to be true.

□

B.1.3 Proof of Lemma 3

Proof. Consider two binary vectors $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ that differ in at least one bit. Without loss of generality, assume that the first bit is one of the bits that they differ in. Let $z_{i,k}$ be the k-th element of $\boldsymbol{z}_i$ and similarly let $z_{j,k}$ be the k-th element of $\boldsymbol{z}_j$. We can further assume without loss of generality that $z_{i,1} = 0$ and $z_{j,1} = 1$. Then, for any Bernoulli(0.5) vector $\mathbf{b}$, we have

$$\begin{aligned}
\langle \boldsymbol{z}_i, \mathbf{b} \rangle_\oplus &= (0 \cdot \mathrm{b}_1) \oplus (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}), \\
\langle \boldsymbol{z}_j, \mathbf{b} \rangle_\oplus &= (1 \cdot \mathrm{b}_1) \oplus (z_{j,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{j,d'} \cdot \mathrm{b}_{d'}),
\end{aligned}$$

where b_k is the k-th element of $\mathbf{b}$. There can be two cases:

- $(z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}) = (z_{j,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{j,d'} \cdot \mathrm{b}_{d'})$. Say the probability of this is p. Note that b_1 is independent of every other b_i. With probability 0.5, $\mathrm{b}_1 = 1$. In this case,

$$\begin{aligned}
\langle \boldsymbol{z}_i, \mathbf{b} \rangle_\oplus &= (0 \cdot 1) \oplus (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}) \\
&= 0 \oplus (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}) \\
&= (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}) \\
&= \neg(\neg((z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}))) \\
&\qquad \text{(Applying two NOT operations does not affect the value.)} \\
&= \neg(1 \oplus (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'})) \\
&\qquad \text{(XOR with 1 is equivalent to the NOT operation.)} \\
&= \neg((1 \cdot 1) \oplus (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'})) \\
&= \neg\langle \boldsymbol{z}_j, \mathbf{b} \rangle_\oplus.
\end{aligned}$$

  Hence, in this case, with probability at least 0.5, $\langle \boldsymbol{z}_i, \mathbf{b} \rangle_\oplus \neq \langle \boldsymbol{z}_j, \mathbf{b} \rangle_\oplus$.

- $(z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}) = \neg((z_{j,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{j,d'} \cdot \mathrm{b}_{d'}))$. The probability of this is $1 - p$. Note that b_1 is independent of every other b_i. With probability 0.5, $\mathrm{b}_1 = 0$. In this case,

$$\begin{aligned}
\langle \boldsymbol{z}_i, \mathbf{b} \rangle_\oplus &= (0 \cdot 0) \oplus (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}) \\
&= 0 \oplus (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}) \\
&= (z_{i,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{i,d'} \cdot \mathrm{b}_{d'}) \\
&= \neg((z_{j,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{j,d'} \cdot \mathrm{b}_{d'})) \\
&= \neg((1 \cdot 0) \oplus (z_{j,2} \cdot \mathrm{b}_2) \oplus \cdots \oplus (z_{j,d'} \cdot \mathrm{b}_{d'})) \\
&= \neg\langle \boldsymbol{z}_j, \mathbf{b} \rangle_\oplus
\end{aligned}$$

  Hence, in this case as well, with probability at least 0.5, $\langle \boldsymbol{z}_i, \mathbf{b} \rangle_\oplus \neq \langle \boldsymbol{z}_j, \mathbf{b} \rangle_\oplus$.

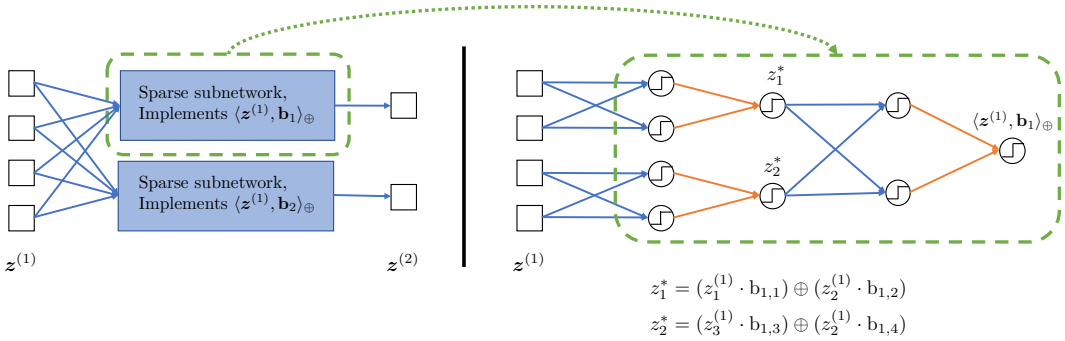

Figure 5: **Left:** We can compute each of $\langle \boldsymbol{z}^{(1)}, \mathbf{b}_k \rangle$ using subnetworks shown in blue, that we can put in parallel to output $\boldsymbol{z}^{(2)}$. **Right:** A subnetwork that implements $\langle \boldsymbol{z}^{(1)}, \mathbf{b}_1 \rangle$. This, in turn can be broken down as $\langle \boldsymbol{z}^{(1)}, \mathbf{b}_1 \rangle_\oplus = (z_1^{(1)} \cdot \mathrm{b}_{1,1}) \oplus (z_2^{(1)} \cdot \mathrm{b}_{1,2}) \oplus \cdots \oplus (z_{d^{(1)}}^{(1)} \cdot \mathrm{b}_{1,d^{(1)}})$. Then, the figure on the right shows how the network computes $\langle \boldsymbol{z}^{(1)}, \mathbf{b}_1 \rangle$ using a binary tree structure (see the orange lines in the figure). In the figure, we take $d^{(1)} = 4$. The network first computes $z_1^* = (z_1^{(1)} \cdot \mathrm{b}_{1,1}) \oplus (z_2^{(1)} \cdot \mathrm{b}_{1,2})$ and $z_2^* = (z_3^{(1)} \cdot \mathrm{b}_{1,3}) \oplus (z_4^{(1)} \cdot \mathrm{b}_{1,4})$. Then, it computes $z_1^* \oplus z_2^*$, as shown in the figure.

Hence overall, with probability at least $p(0.5)+(1-p)(0.5) = 0.5$, we have that $\langle \boldsymbol{z}_i, \mathbf{b} \rangle_\oplus \neq \langle \boldsymbol{z}_j, \mathbf{b} \rangle_\oplus$. If we take m i.i.d. Bernoulli(0.5) vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$, then the probability that

$$\forall k \in [m], \langle \boldsymbol{z}_i, \mathbf{b}_k \rangle_\oplus = \langle \boldsymbol{z}_j, \mathbf{b}_k \rangle_\oplus,$$

is less than $(0.5)^m$. Equivalently,

$$\Pr(\boldsymbol{z}_i^{(2)} = \boldsymbol{z}_j^{(2)}) \leq 2^{-m}.$$

To get the bound required in the lemma statement, we simply take a union bound over all the $\binom{n}{2}$ pairs $(\boldsymbol{z}_i^{(2)}, \boldsymbol{z}_j^{(2)})$. Concretely, we want $\Pr(\exists i, j : \boldsymbol{z}_i^{(2)} = \boldsymbol{z}_j^{(2)}) \leq \epsilon$. Hence, we take a union bound:

$$\begin{aligned}\Pr(\exists i, j : \boldsymbol{z}_i^{(2)} = \boldsymbol{z}_j^{(2)}) &\leq \binom{n}{2} \Pr(\boldsymbol{z}_i^{(2)} = \boldsymbol{z}_j^{(2)}) \\ &\leq \binom{n}{2} 2^{-m}.\end{aligned}$$

It can be easily verified that $m \geq 3 \log \frac{n}{\epsilon}$ suffices for $\binom{n}{2} 2^{-m} \leq \epsilon$. This completes the proof. □

B.1.4 Proof of Lemma 4

Proof. We will show that we can compute $\langle \boldsymbol{z}_i^{(1)}, \mathbf{b}_k \rangle_\oplus$ with a network of $3(d^{(1)} - 1)$ neurons and $9(d^{(1)} - 1)$ weights. Then, we can put m such networks in parallel (see Figure5, Left) to output

$$\boldsymbol{z}_i^{(2)} = \begin{bmatrix} \langle \mathbf{b}_1, \boldsymbol{z}_i \rangle_\oplus \\ \vdots \\ \langle \mathbf{b}_m, \boldsymbol{z}_i \rangle_\oplus \end{bmatrix}.$$

Hence, all we need to do is to prove that we can compute $\langle \boldsymbol{z}_i^{(1)}, \mathbf{b}_k \rangle_\oplus$ with a network of $3(d^{(1)} - 1)$ neurons and $9(d^{(1)} - 1)$ weights. We will do this in a hierarchical fashion as shown in Figure 5, Right. We begin by expanding $\langle \boldsymbol{z}_i^{(1)}, \mathbf{b}_k \rangle_\oplus$:

$$\langle \boldsymbol{z}_i^{(1)}, \mathbf{b}_k \rangle_\oplus = (z_{i,1}^{(1)} \cdot \mathrm{b}_{k,1}) \oplus (z_{i,2}^{(1)} \cdot \mathrm{b}_{k,2}) \oplus \cdots \oplus (z_{i,d^{(1)}}^{(1)} \cdot \mathrm{b}_{k,d^{(1)}}).$$

Note that

$$(z_{i,1}^{(1)} \cdot \mathrm{b}_{k,1}) \oplus (z_{i,2}^{(1)} \cdot \mathrm{b}_{k,2}) = \sigma(\sigma((z_{i,1}^{(1)} \cdot \mathrm{b}_{k,1}) - (z_{i,2}^{(1)} \cdot \mathrm{b}_{k,2}) - 1) + \sigma(-(z_{i,1}^{(1)} \cdot \mathrm{b}_{k,1}) + (z_{i,2}^{(1)} \cdot \mathrm{b}_{k,2}) - 1))),$$

which uses 3 neurons, each with 2 weights and a bias. Similarly, we can compute $(z_{i,3}^{(1)} \cdot \mathrm{b}_{k,3}) \oplus (z_{i,4}^{(1)} \cdot \mathrm{b}_{k,4})$, $(z_{i,5}^{(1)} \cdot \mathrm{b}_{k,5}) \oplus (z_{i,6}^{(1)} \cdot \mathrm{b}_{k,6}), \ldots, (z_{i,d^{(1)}-1}^{(1)} \cdot \mathrm{b}_{k,d^{(1)}-1}) \oplus (z_{i,d^{(1)}}^{(1)} \cdot \mathrm{b}_{k,d^{(1)}})$. Once we have computed these, we can compute XOR's of four terms similarly, to get terms like $(z_{i,1}^{(1)} \cdot \mathrm{b}_{k,1}) \oplus (z_{i,2}^{(1)} \cdot \mathrm{b}_{k,2}) \oplus$. We keep applying this hierarchically in a binary tree form to get $\langle \boldsymbol{z}_i, \mathbf{b}_k \rangle_{\oplus} (z_{i,3}^{(1)} \cdot \mathrm{b}_{k,3}) \oplus (z_{i,4}^{(1)} \cdot \mathrm{b}_{k,4})$ (see Figure 5, Right). The tree will consist of $d^{(1)} - 1$ XOR operations, each represented by a node. For each of these operations, we saw above that we need 3 neurons, each with 2 weights and a bias. This gives that we need $3(d^{(1)} - 1)$ neurons and $9(d^{(1)} - 1)$ weights and biases to compute $\langle \boldsymbol{z}_i^{(1)}, \mathbf{b}_k \rangle_{\oplus}$.

□

B.1.5 Proof of Lemma 5

Proof. Consider a binary tree of depth $d^{(2)} + 1$, were the left child edge of each node is labeled 0, and the right one is labeled 1. There will be $2^{d^{(2)}}$ leaves, each representing one unique binary vector of length $d^{(2)}$. The vector associated with a leaf would be the same as the sequence of edges of the path from the root to that leaf. Hence, $\{\boldsymbol{z}_1^{(2)}, \ldots, \boldsymbol{z}_n^{(2)}\}$ will form a subset of the $2^{d^{(2)}}$ leaves. Remove all other leaves from the tree, so that we only have $\{\boldsymbol{z}_1^{(2)}, \ldots, \boldsymbol{z}_n^{(2)}\}$ as the leaves in our tree. Hence, now our tree only has n leaves. Also remove all the nodes that are not the ancestor of any leaves.

The way we select subsets is simple: Each node is a potential subset. Each node represents the subset of $\{\boldsymbol{z}_1^{(2)}, \ldots, \boldsymbol{z}_n^{(2)}\}$ that belong in the subtree at that node. Each node is also associated with a unique prefix: the sequence of edges of the path from the root to that node. Denote by $\mathcal{S}$ the set of nodes (or equivalently subsets) that we select. We include a node in $\mathcal{S}$, if

1. the number of leaves in the subtree at that node is at most $\sqrt{n}$,
2. none of its ancestor nodes satisfy condition 1.

Condition 1 will ensure that the subsets formed in this way will have size at most $\sqrt{n}$. Condition 2 above ensures that the subsets do not overlap and consequently the prefixes of the selected nodes are unique. Hence, all that is left to do is prove that $|\mathcal{S}| = \mathcal{O}(\sqrt{n} \log n)$.

Define the height of a node as ($d^{(2)}$ − its depth). Thus, the root node has height $d^{(2)}$ and the leaves have height 0. We will prove that for any node at height h, if any of its descendant nodes is chosen, then

$$\frac{\#\text{ leaves in the subtree at that node}}{\#\text{ descendant nodes included in } \mathcal{S}} \geq \frac{\sqrt{n}}{h}. \tag{7}$$

At root node, $h = d^{(2)} = c \log n$, and $\#$ leaves $= n$. Then, rearranging the inequality (7), we get that at the root node the number of descendant nodes included in $\mathcal{S}$ are less than $\sqrt{n}(c \log n)$. Thus, the total number of subsets is $\mathcal{O}(\sqrt{n} \log n)$. Hence, all we need to do now, is to prove (7). For this, we will use induction on h.

The smallest h for which the inequality is applicable, will be $h = \lceil \log_2 \sqrt{n} \rceil +$. This is because the first height at which a node can be selected is $\lceil \log_2 \sqrt{n} \rceil$. Note that no descendants of a node at height $h = \lceil \log_2 \sqrt{n} \rceil + 1$, other than its children, can be included in $\mathcal{S}$. Further, since one of its children is included in $\mathcal{S}$, this node itself is not included in $\mathcal{S}$. Hence, the number of leaves in the subtree at this node is at least $\sqrt{n} + 1$. Hence for this node,

$$\frac{\#\text{ leaves in the subtree formed at that node}}{\#\text{ descendant nodes in } \mathcal{S}} \geq \frac{\sqrt{n}+1}{2} \geq \frac{\sqrt{n}}{h}.$$

Thus, the base case is proved. We move on to the inductive case now: $h > \lceil \log_2 \sqrt{n} \rceil + 1$. There can be the following cases for a node s at height h.

- s has only one child. This child cannot be in $\mathcal{S}$, since Condition 2 would be violated. Thus, it can only be the case that this child has further descendants that are in $\mathcal{S}$. Then, by the induction hypothesis, (7) is satisfied for the child, and hence it is also satisfied for s.
- Both of its children have descendants that are in $\mathcal{S}$. Let l_{l} be the ($\#$ leaves in the subtree formed at the left child of s) and d_{l} be the ($\#$ descendant nodes of the left child in $\mathcal{S}$). Similarly, we define l_{r} and d_{r} for the right child. Then, we know by the induction hypothesis that

$$\frac{l_{\text{l}}}{d_{\text{l}}} \geq \frac{\sqrt{n}}{h-1}, \frac{l_{\text{r}}}{d_{\text{r}}} \geq \frac{\sqrt{n}}{h-1}.$$

  Hence, for the node s,

$$\begin{aligned}\frac{\#\text{ leaves in the subtree formed at } s}{\#\text{ descendant nodes in } \mathcal{S}} &= \frac{l_{\text{l}} + l_{\text{r}}}{d_{\text{l}} + d_{\text{r}}} \\ &\geq \frac{\frac{\sqrt{n}}{h-1}(d_{\text{l}} + d_{\text{r}})}{d_{\text{l}} + d_{\text{r}}} \\ &\geq \frac{\sqrt{n}}{h}.\end{aligned}$$

- One of its children is included in $\mathcal{S}$ and the other node has at least one of its descendants included in $\mathcal{S}$. Let the left child be the one that is included in $\mathcal{S}$. Let l_{r} be the ($\#$ leaves in the subtree formed at the right child of s) and d_{r} be the ($\#$ descendant nodes of the right child included in $\mathcal{S}$). Then, by inductive hypothesis,

$$\frac{l_{\text{r}}}{d_{\text{r}}} \geq \frac{\sqrt{n}}{h-1}.$$

  We also know that since the right child wasn't included in $\mathcal{S}$, $l_{\text{r}} > \sqrt{n}$. Note that the number of leaves under s is at least l_r and the number of its descendants that are in $\mathcal{S}$ is $d_{\text{r}} + 1$, including the left child. Then,

$$\begin{aligned}\frac{\#\text{ leaves in the subtree formed at } s}{\#\text{ descendant nodes in } \mathcal{S}} &\geq \frac{l_{\text{r}}}{1 + d_{\text{r}}} \\ &= \frac{1}{\frac{1}{l_{\text{r}}} + \frac{d_{\text{r}}}{l_{\text{r}}}} \\ &\geq \frac{1}{\frac{1}{l_{\text{r}}} + \frac{h-1}{\sqrt{n}}} \\ &= \frac{\sqrt{n}}{\frac{\sqrt{n}}{l_{\text{r}}} + h - 1} \\ &> \frac{\sqrt{n}}{h}.\end{aligned}$$

- Both the children of the node are in $\mathcal{S}$. In this case, since the node itself wasn't chosen, the number of leaves in the subtree is at least $\sqrt{n} + 1$, and the number of descendant nodes in $\mathcal{S}$ is 2. Hence,

$$\frac{\#\text{ leaves in the subtree formed by that node}}{\#\text{ descendant nodes in } \mathcal{S}} \geq \frac{\sqrt{n} + 1}{2} \geq \frac{\sqrt{n}}{h}.$$

This proves the inductive case and completes the proof of lemma. □

B.2 Constructing a shallow network for memorization

The network constructed in this proof has depth $O(\log \frac{\log n}{\delta})$, which is due to the sparse subnetwork which compresses the $\mathcal{O}(\frac{\log n}{\delta})$ sized binary representation down to $\mathcal{O}(\log n)$ sized

binary representation. Removing this subnetwork will bring the depth of the resulting network down to 4 hidden layers. However, doing this will also increase the number of neurons needed from $\mathcal{O}(\frac{\log^2 n}{\delta} + \sqrt{n}\log^2 n)$ to $\mathcal{O}(\frac{\sqrt{n}\log^2 n}{\delta})$, and the number of weights needed from $\mathcal{O}(\frac{(d+\log n)\log n}{\delta} + n\log^2 n)$ to $\mathcal{O}(\frac{(d+n)\log^2 n}{\delta} + \frac{\sqrt{n}\log^2 n}{\delta^2})$.

B.3 Comments on generalization

We remark here that the construction presented in this section does not imply any results on the generalization capacity of the network. However, note that in essence, the first layer partitions the space into convex polytopes and the rest of the network assigns labels to these polytopes. Making the first layer wider would result in a finer partition of the space, into smaller polytopes. Each point within a polytope would have the same binary representation after the first layer, and hence the rest of the network can assign different labels to different polytopes. Finally, if the first layer is wide enough, we can have a very fine partition of the space, and approximate the nearest neighbor classifier. This can lead to guarantees on generalization. Exploring this would be an interesting direction of future research.

C Proof of Theorem 2 and Theorem 5

Proof. We want to construct an example where there are n points on the unit sphere $\mathcal{S}^{d-1}$, which are separated by a distance at least δ, such that the minimum number of hyperplanes needed for separating each pair is $\Omega(\log(n)/\delta)$ and for separating pairs with opposite labels is $\Omega(1/\delta)$.

Before we start the proof, let us define some notation. We denote by $\mathcal{S}^{d-1} := \{x : \|x\| = 1\}$, the unit sphere in d-dimensions centered at the origin. We can also have lower dimensional spheres in the same space. For example, we can have circles in a 3-dimensional space. We denote by $\mathcal{S}^{d-2}_{\boldsymbol{c}}$, the $d-1$ dimensional unit sphere, centered at $\boldsymbol{c}$ and which lies on the hyperplane orthogonal to the vector $\boldsymbol{c}$. For example, in 3-dimensions, let $\boldsymbol{c} = (1,1,1)$. Let $\mathcal{H}_{\boldsymbol{c}}$ be the hyperplane that passes through $\boldsymbol{c}$ and is orthogonal to the vector connecting the origin to $\boldsymbol{c}$. On this hyperplane, we can draw a unit circle, centered at $(1,1,1)$, which is what we denote by $\mathcal{S}^2_{(1,1,1)}$ in our notation. Further, $r\mathcal{S}^{d-1}$ and $r\mathcal{S}^{d-2}_{\boldsymbol{c}}$ will denote the corresponding hyperspheres, but with radius r instead of 1. We denote the cluster centered at $\boldsymbol{c}$ by $\mathcal{Q}_{\boldsymbol{c}}$. We will use $C_1, C_2, \ldots$ to denote universal constants.

Now, we are ready to start the construction. We will create $\sqrt{n}$ clusters of points on the sphere, each of size $\sqrt{n}$.

First, the centers of the $\sqrt{n}$ clusters are sampled uniformly from $(\sqrt{1-\delta})\mathcal{S}^{d-1}$. Then, in each cluster, $\sqrt{n}$ points are sampled i.i.d. from $\sqrt{\delta}\mathcal{S}^{d-2}_{\boldsymbol{c}}$, where $\boldsymbol{c}$ is the center of the cluster. This way, each point has norm $\sqrt{(\sqrt{1-\delta})^2 + (\sqrt{\delta})^2} = 1$, that is, each point indeed lies on $\mathcal{S}^d$. Of the $\sqrt{n}$ points in each cluster, we label half of them as 0 and the other half as 1.

Then, roughly speaking, the summary of the proof is as follows:

- Any hyperplane can only 'effectively' separate points within a cluster if it passes close to the center of the cluster. The distance would roughly need to be $\widetilde{O}(\sqrt{\delta}/\sqrt{d})$.
- A hyperplane can be $\widetilde{O}(\sqrt{\delta}/\sqrt{d})$ close to the centers of only $\widetilde{O}(\sqrt{\delta})$ fraction of the clusters.
- Any cluster needs at least
  - $\Omega(\log_2 \sqrt{n}) = \Omega(\log n)$ hyperplanes to separate each pair of points within the cluster (for Theorem 5) or
  - at least 1 hyperplane to separate all the opposite label pairs (for Theorem 2).
- Finally, the core inequality that we use is that

$$\begin{aligned}(\#\text{ hyperplanes needed}) \times (\#\text{ clusters a hyperplane can 'effectively' separate}) \\ \geq (\#\text{ clusters}) \times (\#\text{ hyperplanes needed per cluster}).\end{aligned} \tag{8}$$

Substituting the bounds we got above into this inequality, we get that we need at least $\Omega(\log(n)/\sqrt{\delta})$ hyperplanes.

Now we describe the construction in detail. Note that we only need to prove the existence of *one* such construction. Thus, we will use the probabilistic method.

We define four events:

$\mathcal{E}_1$: Two points from different clusters have distance less than δ.

$\mathcal{E}_2$: The points within a cluster have distance less than δ.

$\mathcal{E}_3$: A hyperplane at distance $\Omega(\sqrt{\frac{\delta \log t}{d}})$ from the center of a cluster separates $\Omega(n/t)$ pairs of points in a cluster, for $t \geq 4$.

$\mathcal{E}_4$: A hyperplane passes within distance $O(\sqrt{\frac{\delta \log t}{d}})$ of the centers of $\Omega(\sqrt{\delta n \log t})$ clusters.

We will prove that the probability of each of these events is at most $1/5$. Hence, a union bound shows that there exists an event where none of these events happen. We will prove that such an event gives the required lower bounds.

Step 1: Bound the probability of $\mathcal{E}_1$. Let $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{\sqrt{n}}$ be $\sqrt{n}$ cluster centers sampled uniformly from $(\sqrt{1-\delta})\mathcal{S}^{d-1}$, representing the centers of the clusters. Then there exists a universal constant C, such that, as long as $n < \left(\frac{C}{\sqrt{\delta}}\right)^{d-1}$, we show that the event that two points from different clusters have distance less than δ will have probability less than $1/5$.

To see how this is true, consider any two clusters, with centers $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$. Then, as long as $\|\boldsymbol{c}_i - \boldsymbol{c}_j\| \geq 2\sqrt{\delta} + \delta$, we get by triangle inequality that two points from these two clusters are at least δ distance apart. Since $\sqrt{\delta} \geq \delta$, we know that if we ensure that all cluster centers are at least $3\sqrt{\delta}$ apart, then the above still holds.

Hence, we need to show that if we sample $\sqrt{n}$ points (cluster centers) on a sphere of radius $\sqrt{1-\delta}$, and $n < \frac{1}{5}\left(\frac{C}{\sqrt{\delta}}\right)^{d-1}$, then they are at least $3\sqrt{\delta}$ apart with probability at least $4/5$. For this, we need to show that the probability of any pair of points $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ being within $3\sqrt{\delta}$ distance is less than $\left(\frac{\sqrt{\delta}}{C}\right)^{d-1}$. Taking a union bound over all the $\binom{\sqrt{n}}{2}$ pairs and substituting $n < \left(\frac{C}{\sqrt{\delta}}\right)^{d-1}$ will give that the probability of any such pair existing is less than $1/5$.

Thus, all we need to do is to prove that the probability of any pair of points being within $3\sqrt{\delta}$ distance is less than $\left(\frac{\sqrt{\delta}}{C}\right)^{d-1}$. We prove it below.

Fix the point $\boldsymbol{c}_i$. Then, the probability of any other cluster center $\boldsymbol{c}_j$ being within distance $3\sqrt{\delta}$ of $\boldsymbol{c}_i$ is the same as the ratio of the surface area of the cap of radius $3\sqrt{\delta}$ to the surface area of $\sqrt{1-\delta}\mathcal{S}^{d-1}$. Consider packing the surface of $\sqrt{1-\delta}\mathcal{S}^{d-1}$ with caps of radius $3\sqrt{\delta}$, and let $\mathcal{A}_{3\sqrt{\delta}}$ denote such a cap. Let the number of caps required for the packing be $\mathcal{P}'$. Then,

$$\begin{aligned} &\text{Area}(3\sqrt{\delta}\text{ cap}) \times \mathcal{P}' \leq \text{Area}(\sqrt{1-\delta}\mathcal{S}^{d-1}) \\ \implies &\frac{\text{Area}(3\sqrt{\delta}\text{ cap})}{\text{Area}(\sqrt{1-\delta}\mathcal{S}^{d-1})} \leq \frac{1}{\mathcal{P}'}. \end{aligned}$$

Further, note that by rescaling $|\mathcal{P}'|$ is equal to $|\mathcal{P}_{3\sqrt{\delta}/\sqrt{1-\delta}}|$, where $\mathcal{P}_{\delta'}$ is the δ' packing of $\mathcal{S}^{d-1}$. Finally, we use the following lemma (proved later in the appendix) to conclude that the probability of any pair of points being within $3\sqrt{\delta}$ distance is less than $\left(\frac{\sqrt{\delta}}{C}\right)^{d-1}$, for some universal constant C.

Lemma 6. *Let $\mathcal{C}_\delta$ and $\mathcal{P}_\delta$ denote the covering and packing numbers of $\mathcal{S}^{d-1}$, respectively. Then,*

$$\left(\frac{1}{4\delta}\right)^{d-1} \leq \mathcal{C}_\delta \leq \mathcal{P}_\delta \leq \left(\frac{2}{\delta}\right)^{d}.$$

Step 2: Bound the probability of $\mathcal{E}_2$. Consider any one cluster center $\boldsymbol{c}$. Then, the sphere $\sqrt{\delta}\mathcal{S}_{\boldsymbol{c}}^{d-2}$ is just a $d-2$ dimensional sphere. We create this cluster by sampling points uniformly on $\mathcal{S}_{\boldsymbol{c}}^{d-2}$. This is similar to what we did in previous step and hence we can apply exactly the same steps as above to get that as long as $n < \left(\frac{C}{\sqrt{\delta}}\right)^{d-2}$, we have that the event that two points within some cluster have distance less than δ, will have probability less than $1/5$. We denote this cluster by $\mathcal{Q}_{\boldsymbol{c}}$. Then, $\mathcal{Q}_{\boldsymbol{c}}$ lies on the sphere $\sqrt{\delta}\mathcal{S}_{\boldsymbol{c}}^{d-2}$, which in turn lies on the hyperplane $\mathcal{H}_{\boldsymbol{c}}$.

We need to create such clusters around each of the other cluster centers. A trick that we employ is that we use the same cluster $\mathcal{Q}_{\boldsymbol{c}}$ and 'paste' it at every other cluster center, after rotating appropriately to make the cluster lie on the hyperplane orthogonal to the corresponding cluster center. This is helpful because we would not have to take a union bound of the probability calculation done above for every cluster. We formally describe the procedure below.

Let $\boldsymbol{c}'$ be another cluster center. Then, there will be a corresponding sphere $\sqrt{\delta}\mathcal{S}_{\boldsymbol{c}'}^{d-2}$, which lies on the hyperplane $\mathcal{H}_{\boldsymbol{c}'}$. Let $T_{c'}$ be any linear transformation such that $T(\sqrt{\delta}\mathcal{S}_{\boldsymbol{c}}^{d-2}) = \sqrt{\delta}\mathcal{S}_{\boldsymbol{c}'}^{d-2}$. Then, we create the cluster $\mathcal{Q}_{\boldsymbol{c}'}$ for the cluster center $\boldsymbol{c}'$ as $\mathcal{Q}_{\boldsymbol{c}'} = T(\mathcal{Q}_{\boldsymbol{c}})$.

Step 3: Bound the probability of $\mathcal{E}_3$. Any hyperplane $\mathcal{H}_2$ (distinct from $\mathcal{H}_{\boldsymbol{c}}$) that intersects $\mathcal{H}_{\boldsymbol{c}}$ creates a $d-1$ dimensional hyperplane, say $\mathcal{H}_2'$. We want to show that if $\mathcal{H}_2$ lies at distance $\Omega(\sqrt{\frac{\delta \log t}{d}})$ from a cluster center $\boldsymbol{c}$, then the probability that it separates $\geq n/t$ pairs of points in the cluster is less than $1/5$. We will actually show that if $\mathcal{H}_2'$ lies at distance $\Omega(\sqrt{\frac{\delta \log t}{d}})$ from $\boldsymbol{c}$, then the probability that it separates $\Omega(n/t)$ pairs of points in the cluster is less than $1/5$, which is a stronger statement because the distance of $\boldsymbol{c}$ from $\mathcal{H}_2$ is smaller than the distance of $\boldsymbol{c}$ from $\mathcal{H}_2'$. We do not consider hyperplanes parallel to $\mathcal{H}_{\boldsymbol{c}}$ because such hyperplanes do not separate any points in $\mathcal{Q}_{\boldsymbol{c}}$. We will work in the $d-1$ dimensional subspace spanned by $\mathcal{H}_{\boldsymbol{c}}$, with $\boldsymbol{c}$ as the origin.

Take any one direction in this space. The probability measure of $\sqrt{\delta}\mathcal{S}_{\boldsymbol{c}}^{d-2}$ at distance $\geq 2\sqrt{\frac{\delta \log t}{d-1}}$ in this direction is at most $e^{-2\log t}$ (Ball et al., 1997, Lemma 2.2). Thus, the probability that a point in the cluster lies at a distance more than $2\sqrt{\frac{\delta \log t}{d-1}}$ is less than t^{-2}. We can now apply Binomial tail probability bounds to get a probability bound on the event

$$E : \{\text{more than } \frac{1}{t} \text{ fraction of the } \sqrt{n} \text{ points lie at a distance more than } 2\sqrt{\frac{\delta \log t}{d-1}} \text{ in one fixed direction}\}.$$

Let $D(\cdot\|\cdot)$ denote the KL-divergence. Then, the Chernoff bound on the tail probability of a Binomial random variable (Arratia & Gordon, 1989) gives the following bound

$$\begin{aligned}
\Pr(E) &\leq e^{-\sqrt{n}D(\frac{1}{t}\|\frac{1}{t^2})} \\
&= \exp\left(-\sqrt{n}\left(\frac{1}{t}\log t + \left(1-\frac{1}{t}\right)\log\frac{1-\frac{1}{t}}{1-\frac{1}{t^2}}\right)\right) \\
&= \exp\left(-\sqrt{n}\left(\frac{1}{t}\log t + \left(1-\frac{1}{t}\right)\log\frac{t}{1+t}\right)\right) \\
&= \exp\left(-\sqrt{n}\left(\log t - \log(1+t) + \frac{1}{t}\log(1+t+t^2)\right)\right) \\
&= \exp\left(-\sqrt{n}\left(\log\left(\frac{t}{1+t}\right) + \frac{1}{t}\log(1+t)\right)\right) \\
&= \exp\left(-\sqrt{n}\left(\log\left(1-\frac{1}{1+t}\right) + \frac{1}{t}\log(1+t)\right)\right)
\end{aligned}$$

$$
\begin{aligned}
&\leq \exp\left(-\sqrt{n}\left(-\frac{1}{2(1+t)}+\frac{1}{t}\log(1+t)\right)\right) && \text{(Since } \tfrac{1}{1+t}\leq 1\text{)}\\
&\leq \exp\left(\frac{-\sqrt{n}}{2t}\log(1+t)\right) && \text{(Since } t\geq 2\text{)}\\
&\leq \frac{1}{t^{\sqrt{n}/2t}}.
\end{aligned}
$$

Recall that the above result is just for one fixed direction, whereas we want to ensure that no more than $1/t$ fraction of points lie at distance $\Omega(\sqrt{\delta \log t/(d-1)})$ in *any* direction. For this, we need an ϵ-net argument[5] over all the directions. Note that a direction is essentially just a point on the unit sphere and hence, we need to make an ϵ-net on $\mathcal{S}^{d-2}$. We claim that for $\epsilon = \sqrt{\frac{\log t}{d-1}}$, if we prove that no more than $1/t$ fraction of points lie at distance $2\sqrt{\delta \log t/(d-1)}$ in any direction in the ϵ-net, then no more than $1/t$ fraction of points lie at distance $4\sqrt{\delta \log t/(d-1)}$ in any direction. We will prove this claim shortly. Recall that an ϵ-net is the same as a covering and hence we can use Lemma 6 to get an upper bound on the size of the ϵ-net (Note that the radius of the sphere here is $\sqrt{\delta}$, so we need to rescale appropriately). Taking a union bound over the ϵ-net gives the probability

$$
p := \left(4\sqrt{\frac{d-1}{\log t}}\right)^{d-1} \frac{1}{t^{\sqrt{n}/2t}}. \tag{9}
$$

Thus, any hyperplane that lies at distance more than $4\sqrt{\frac{\delta \log t}{d-1}}$ from the center of a cluster has less than $\sqrt{n}/t$ points on one side of it, with probability at least $1-p$. Note that if a hyperplane has less than $\sqrt{n}/t$ points on one side, then it can only separate $\leq \frac{\sqrt{n}}{t}(\sqrt{n}-\frac{\sqrt{n}}{t}) \leq \frac{n}{t}$ pairs of points, which is what we want to prove in this step. Hence, we want to show that $p < 1/5$. From (9), we can see that this will be true as long as $\sqrt{n} \geq C_3 dt \log d$, where C_3 is a large enough universal constant.

To finish this step, we need to prove that an ϵ-net would ensure that in every direction, no more than $1/t$ fraction of points lie beyond $4\sqrt{\frac{\delta \log t}{d-1}}$. For any direction (or equivalently, a unit vector) $\boldsymbol{u}$, there is a direction $\boldsymbol{u}_e$ in our epsilon net such that $\|\boldsymbol{u}-\boldsymbol{u}_e\| \leq 2\sqrt{\frac{\log t}{d-1}}$. Consider the region of $\sqrt{\delta}\mathcal{S}_{\boldsymbol{c}}^{d-2}$ consisting of points that lie at distance $4\sqrt{\frac{\delta \log t}{d-1}}$ from the origin (*i.e.,* $\boldsymbol{c}$) in the direction of $\boldsymbol{u}$. Let $\boldsymbol{x}$ be any point in the region. Then,

$$
\begin{aligned}
\langle \boldsymbol{x}, \boldsymbol{u}_e\rangle &= \langle \boldsymbol{x}, \boldsymbol{u}\rangle + \langle \boldsymbol{x}, \boldsymbol{u}_e - \boldsymbol{u}\rangle\\
&\geq 4\sqrt{\frac{\delta \log t}{d-1}} + \langle \boldsymbol{x}, \boldsymbol{u}_e - \boldsymbol{u}\rangle\\
&\geq 4\sqrt{\frac{\delta \log t}{d-1}} - \|\boldsymbol{x}\|\|\boldsymbol{u}_e - \boldsymbol{u}\|\\
&\geq 4\sqrt{\frac{\delta \log t}{d-1}} - 2\sqrt{\frac{\delta \log t}{d-1}}\\
&= 2\sqrt{\frac{\delta \log t}{d-1}}.
\end{aligned}
$$

Hence, any point in such a region lies at distance $\geq 2\sqrt{\frac{\delta \log t}{d-1}}$ in the direction $\boldsymbol{u}_e$.

Step 4: Bound the probability of $\mathcal{E}_4$. This is similar to the previous step. We need to show that there are only $\mathcal{O}(\sqrt{\delta n \log t})$ cluster centers within distance $\mathcal{O}(\sqrt{\delta \log t/d})$ of any hyperplane.

Take any fixed hyperplane. Then, the probability measure of the sphere $\sqrt{1-\delta}\mathcal{S}^{d-1}$ within distance $12\sqrt{\delta \log t}/\sqrt{d}$ of the hyperplane is less than $C_5\sqrt{\delta \log t}$, for some universal constant C_5. Using

[5] An ϵ-net $\mathcal{E}$ of a set $\mathcal{B}$ is a set of points from $\mathcal{B}$ such that any point in $\mathcal{B}$ is at distance at most ϵ from some point in $\mathcal{E}$.

this, we get that the probability that any cluster center lies within distance $12\sqrt{\delta \log t}/\sqrt{d}$ of the hyperplane is less than $C_5\sqrt{\delta \log t}$. Similar to the previous step, we can now use Chernoff bounds to get the probability that more than $2C_5\sqrt{\delta \log t}$ fraction of cluster centers lie at a distance less than $12\sqrt{\delta \log(t)/d}$ is less than $e^{-C_5\sqrt{n\delta \log t}}$.

Recall that this is for a fixed hyperplane, whereas we want to give a probability bound for all hyperplanes. Hence, similar to the previous step, we take a union bound over an ϵ-net of hyperplanes. We claim that if $\epsilon = 4\sqrt{\delta \log(t)/d}$, then the ϵ-net argument will ensure that for any hyperplane, the probability that more than $2C_5\sqrt{\delta \log t}$ fraction of cluster centers lie at a distance less than $4\sqrt{\delta \log(t)/d}$ is less than $e^{-C_5\sqrt{n\delta \log t}}$. We will prove this shortly. We will see that the size of the net will be $\frac{1}{\epsilon}$ times the the covering number of $\mathcal{S}^{d-1}$. Taking a union bound over the ϵ-net gives the probability

$$p' := \frac{1}{\epsilon}\left(C_6\sqrt{\frac{d}{\delta \log t}}\right)^{d-1} e^{-C_5\sqrt{n\delta \log t}}.$$

p' will be less than $1/5$ when $\sqrt{n} > C_7 \frac{d}{\sqrt{\delta}} \log \frac{d}{\delta}$.

Next, we prove that if we create an ϵ-net of hyperplanes, then that would indeed be sufficient to ensure that for any hyperplane, the probability that more than $2C_5\sqrt{\delta \log t}$ fraction of cluster centers lie at a distance less than $4\sqrt{\delta \log(t)/d}$ is less than $e^{-C_5\sqrt{n\delta \log t}}$. A hyperplane can be represented by $h(\boldsymbol{x}) = \boldsymbol{u}^\top \boldsymbol{x} + b$, where $\boldsymbol{u}$ is a unit vector. Similar to previous step, create an ϵ-net over the unit vectors $\boldsymbol{u}$. We also create an ϵ-net over b. However, since we are only interested in hyperplanes that intersect the sphere, we restrict $b \in [0, 1]$, and hence the epsilon net over b would be of size $1/\epsilon$. We know that there is a hyperplane $h_e(\boldsymbol{x}) = \boldsymbol{w}_e^\top \boldsymbol{x} + b_e$ in our ϵ-net such that $\|\boldsymbol{w}_e - \boldsymbol{w}\| \leq \epsilon$ and $|b - b_e| \leq \epsilon$. Consider the region of $\sqrt{1-\delta}\mathcal{S}^{d-1}$ within distance $4\sqrt{\delta \log(t)/d}$ of h. Hence, for any point $\boldsymbol{x}$ in the region,

$$\begin{aligned}
\boldsymbol{w}_e^\top \boldsymbol{x} + b_e &= (\boldsymbol{w}^\top \boldsymbol{x} + b) + (\boldsymbol{w}_e - \boldsymbol{w})^\top \boldsymbol{x} + (b_e - b) \\
&\leq (\boldsymbol{w}^\top \boldsymbol{x} + b) + (\boldsymbol{w}_e - \boldsymbol{w})^\top \boldsymbol{x} + \epsilon \\
&\leq (\boldsymbol{w}^\top \boldsymbol{x} + b) + \|\boldsymbol{w}_e - \boldsymbol{w}\|\|\boldsymbol{x}\| + \epsilon \\
&\leq (\boldsymbol{w}^\top \boldsymbol{x} + b) + \epsilon + \epsilon \\
&\leq 12\sqrt{\delta \log(t)/d}.
\end{aligned}$$

Hence, any such point lies within distance $12\sqrt{\delta \log(t)/d}$.

Step 5: Using the core inequality. We showed in the steps above that with probability at least $1/5$, none of the events $\mathcal{E}_1, \mathcal{E}_1, \mathcal{E}_1, \mathcal{E}_4$ happen. We will work in such an event, ***i.e.,*** the event when none of $\mathcal{E}_1, \mathcal{E}_1, \mathcal{E}_1, \mathcal{E}_4$ happen.

Let h denote the minimum number of hyperplanes needed to separate the dataset that we have constructed. We set $t = 8h$. Then, we showed in Step 3 that any hyperplane at distance more than $4\sqrt{\delta \log(t)/(d-1)}$ from a cluster center can only separate $n/t = n/8h$ pairs of points, whereas there are a total of $n/4$ pairs of opposite label points in each cluster. Hence, we need at least one hyperplane to pass within distance $4\sqrt{\delta \log(t)/(d-1)}$ of the cluster center, otherwise there would be at least $n/8$ pairs of opposite label points that would not be separated. On the other hand, we showed in Step 4 that a hyperplane can pass within distance $4\sqrt{\delta \log(t)/(d-1)}$ of only $2C_5\sqrt{\delta \log t}$ fraction of the cluster centers. Now, we are ready to use the core inequality (8) to get that

$$h \geq \frac{\sqrt{n} \times 1}{2C_5\sqrt{\delta \log t}\sqrt{n}}.$$

This gives that $h = \Omega(1/(\sqrt{\delta}\log(1/\delta)))$.

For the case when we need to separate every pair of points (Theorem 5), the only difference in the inequality above is that we would need at least $\log_2(n - \frac{n}{8})$ hyperplanes per cluster (instead of 1) that need to pass within distance $4\sqrt{\delta \log(t)/(d-1)}$ of the cluster center. This is because separating m points needs $\log_2 m$ hyperplanes. Substituting this in the inequality (8) gives the bound in Theorem 5. □

C.1 Proof of Lemma 6

Proof. Let $\mathcal{P}_\delta$ be a δ packing of the unit sphere $\mathcal{S}^{d-1}$. Let $\mathcal{B}^d$ denote the unit ball, *i.e.,* the sphere $\mathcal{S}^{d-1}$ along with its interior. Then, if we draw balls of radius δ around each point in the packing, they will not intersect and all of them will be completely contained inside the ball of radius $1+\delta$. Thus, we get the following

$$\begin{aligned}
|\mathcal{P}_\delta| \times \text{Vol}(\delta\mathcal{B}^d) &\leq \text{Vol}((1+\delta)\mathcal{B}^d) \\
\implies |\mathcal{P}_\delta| &\leq \frac{\text{Vol}((1+\delta)\mathcal{B}^d)}{\text{Vol}(\delta\mathcal{B}^d)} \\
&= \left(\frac{1+\delta}{\delta}\right)^d \\
&\leq \left(\frac{2}{\delta}\right)^d.
\end{aligned}$$

Next, let $\mathcal{C}_\delta$ be a δ-covering of the sphere $\mathcal{S}^{d-1}$. Any point on the sphere $(1-\delta)\mathcal{S}^{d-1}$ is at distance δ of some point on $\mathcal{S}^{d-1}$. That point is, in turn, at distance within δ of some point in $\mathcal{C}_\delta$. Hence, by the triangle inequality, every point on on the sphere $(1-\delta)\mathcal{S}^{d-1}$ is within distance 2δ of some point in $\mathcal{C}_\delta$. Similarly, it can be shown that every point on on the sphere $(1+\delta)\mathcal{S}^{d-1}$ is within distance 2δ of some point in $\mathcal{C}_\delta$. Draw balls of radius 2δ around each point in the covering $\mathcal{C}_\delta$. Then, these balls cover all the points within distance δ of the sphere $\mathcal{S}^{d-1}$. Thus, we get the following

$$\begin{aligned}
|\mathcal{C}_\delta| \times \text{Vol}(2\delta\mathcal{B}^d) &\geq \text{Vol}(((1+\delta)\mathcal{B}^d) \setminus ((1-\delta)\mathcal{B}^d)), \\
\text{which implies} |\mathcal{C}_\delta| &\geq \frac{\text{Vol}((1+\delta)\mathcal{B}^d) - \text{Vol}((1-\delta)\mathcal{B}^d)}{\text{Vol}(2\delta\mathcal{B}^d)} \\
&= \frac{(1+\delta)^d - (1-\delta)^d}{(2\delta)^d} \\
&\geq \frac{(1+\delta)^d - 1}{(2\delta)^d} \\
&\geq \frac{1+d\delta-1}{(2\delta)^d} \\
&= \frac{d\delta}{(2\delta)^d} \\
&\geq \frac{1}{(4\delta)^(d-1)}.
\end{aligned}$$

Combining with the following inequality (Vershynin, 2018, Lemma 4.2.8):

$$\mathcal{C}_\delta \leq \mathcal{P}_\delta,$$

we get

$$\left(\frac{1}{4\delta}\right)^{d-1} \leq \mathcal{C}_\delta \leq \mathcal{P}_\delta \leq \left(\frac{2}{\delta}\right)^d.$$

□

D Proofs from Section 6

D.1 Proof of Theorem 3

$\mathcal{H}\downarrow, \mathcal{P}\rightarrow$	p_1	p_2	p_3	$\dots$	$p_{\lvert\mathcal{P}\rvert}$
h_1	1	0	0		0
h_2	1	1	1		1
h_3	0	1	0		1
$\vdots$					
$h_{\lvert\mathcal{H}\rvert}$	0	1	0		1

Table 2: (Lower bound) Here, if there are r rows and c columns, and $2^r < c$, then there will be two columns with the same pattern of 0's and 1's. Noting that $r = |\mathcal{H}|$ and $c = |\mathcal{P}|$ gives us the lower bound.

Proof. Let $\mathcal{A}$ output models from the hypothesis set $\mathcal{H}$. Then, we will essentially try to lower bound $\log_2 |\mathcal{H}|$, since we will need at least $\log_2 |\mathcal{H}|$ bits to represent the models.

The first lower bound is easy to see: For a fixed set of n points, there can be 2^n possible assignments of labels. Hence we need at least 2^n hypothesis in our hypothesis class. Hence we need at least n bits to memorize our dataset.

A lower bound with dependence on δ is also not difficult to get. Here is how we create our dataset: Create a δ-packing $\mathcal{P}_\delta$ of the set $\mathcal{S}$. Choose any n points from this packing. In fact, we will only concentrate on the first and the second point in our dataset. Assume that we have a hypothesis set $\mathcal{H}$ such that this can memorize these two points. This means that no matter which two points we choose from the packing and what labels we assign them, there will be a hypothesis in $\mathcal{H}$ that gives the chosen points the assigned labels. Now, if $2^{|\mathcal{H}|} < |\mathcal{P}_\delta|$, then there exist at least one pair of points in the packing $\mathcal{P}_\delta$ such that every hypothesis assigns the two points the same labels. To see how, consider Table 2. There can be at most $2^{|\mathcal{H}|}$ unique columns in that table. Hence, if $2^{|\mathcal{H}|} < |\mathcal{P}_\delta|$, then there would exist two points which have the same columns. We choose those two points and give them opposite labels, then this would contradict memorization, since all the hypotheses in $\mathcal{H}$ give them both the same label. This proves that $2^{|\mathcal{H}|} \geq |\mathcal{P}_\delta|$, or $\log_2 |\mathcal{H}| \geq \log_2 \log_2 |\mathcal{P}_\delta|$. Combined with the first lower bound, we get that we need at least $\max(n, \log_2 \log_2 |\mathcal{P}_\delta|)$ bits. □

D.2 Proof of Theorem 4

$\mathcal{H}\downarrow, \mathcal{C}\rightarrow$	c_1	c_2	c_3	$\dots$	$c_{\lvert\mathcal{C}\rvert}$
h_1	1	0	0		0
h_2	1	1	1		1
h_3	0	1	0		1
$\vdots$					
$h_{\lvert\mathcal{H}\rvert}$	0	1	0		1

Table 3: (Upper bound) Each entry of this table is i.i.d. Bernoulli with probability 0.5. All we need to do is to ensure that if we choose any n columns above and any labeling in $\{0,1\}^n$, there exists one row that has that same labeling.

Proof. Create a $\delta/2$ covering $\mathcal{C}_{\delta/2}$ of the set $\mathcal{S}$. We can partition the sphere into $|\mathcal{C}_{\delta/2}|$ regions of diameter less than δ: Any point that lies within distance $\delta/2$ to any point in $\mathcal{C}_{\delta/2}$ is included in the region of that point. If a point lies within distance $\delta/2$ of multiple points in $\mathcal{C}_{\delta/2}$, then assign it to any one of the regions arbitrarily. Since by assumption, no two points in our dataset are within distance δ, the benefit of this partition is that every pair of points in our dataset will lie in different

regions of the sphere. Consider the set of all the hypotheses h_i that assign the same label to all the points which lie in a region of the partition. (If $\mathcal{S}$ is a sphere, then these hypotheses look like a football with black and white regions: black region is 0, white is 1). There will be $2^{\mathcal{C}_{\delta/2}}$ such hypotheses. We will only select a random subset of this large set and denote it by $\mathcal{H}$. We will show that $\log_2 \mathcal{H} = \mathcal{O}(n + \log_2 \log_2 |\mathcal{C}_{\delta/2}|)$ will suffice for memorization.

For any particular choice of n points and n labels, the probability that none of the $|\mathcal{H}|$ hypothesis have that choice of labels is $(1 - 2^{-n})^{|\mathcal{H}|}$. Taking a union bound over all the $\binom{|\mathcal{C}_{\delta/2}|}{n}$ choices of points and 2^n choices of labels, we need

$$\binom{|\mathcal{C}_{\delta/2}|}{n} 2^n (1 - 2^{-n})^{|\mathcal{H}|} < 1,$$

to ensure that there exists one $\mathcal{H}$ for which we can memorize any n points. Noting that $1 - 2^{-n} < e^{-2^{-n}}$ and $\binom{|\mathcal{C}_{\delta/2}|}{n} < |\mathcal{C}_{\delta/2}|^n$, we see that it is sufficient to have

$$(2|\mathcal{C}_{\delta/2}|)^n e^{-2^{-n}|\mathcal{H}|} < 1.$$

Solving this, we get that $\log_2 |\mathcal{H}| > n + \log_2 n + \log_2 \log_e 2|\mathcal{C}_{\delta/2}|$ is sufficient, *i.e.*, we need $O(n + \log_2 \log_2 |\mathcal{C}_{\delta/2}|)$ bits to memorize n points. □