Swiss Federal Institute of Technology Zurich

Seminar for Statistics

Department of Mathematics

Master Thesis

Summer 2021

Leonardo Kokot

Halfspace learning typical interpolation and 1-layer neural network guarantees

Submission Date: September 10, 2021

Co-Adviser	Felix Kuchelmeister
Adviser:	Prof. Dr. Sara van de Geer

Briefly, I dedicate this work To my family, friends, and all the people that love teChnology, math, and science, and tO all those that do thIngs oNce thought to be impossible.

Preface

I would like to thank Prof. Dr. Sara van de Geer for being my supervisor and making it possible for me to work on such an interesting project as this. Next, I would like to thank Felix Kuchelmeister for being a great mathematician and helping me with his guidance and ideas to finish the project. Finally, I would like to thank Jakov Hrenic for taking a lot of his spare time to read the paper and giving me a lot of help regarding the English grammar and to help me reform and rephrase some parts of the paper that might be very hard to understand for the reader.

Abstract

When giving the guarantees on test set performance of interpolating neural networks, machine learning researchers would traditionally use uniform convergence framework type of bounds. Since these bounds give information on what is the worst performance neural network that you could get, when fitting neural network on a given training data, it has been shown to fail (look into Nagarajan and Kolter (2019)) when applied to complex hypothesis space such as neural networks. This is observed in practice, where we observe that neural networks tend to perform much better than what would be expected by looking at the given guarantees. Therefore, in Theisen, Klusowski, and Mahoney (2021) they try to overcome this issue and look into the distribution of behaviours of random (typical) interpolating neural networks instead. They prove results on convergence of behaviour of typical interpolators as dimension of data grows, but all that under the assumptions which seem to be very unrealistic. Therefore, we build on top of their work and analyze noiseless version of the halfspace learning model (1 layer neural network) under much more realistic assumptions. For training sample size $n = 1$, we prove convergence of behaviour of typical and worst case interpolators as dimension grows to infinity. Typical interpolators converge to a behaviour of a classifier with the error rate of 0.5, while at the same time worst case interpolator converges to having an error rate of 1.0. After that, we support these theoretical findings with simulations under the scenario when the number of training samples n is greater than 1. Finally, we compare results obtained by the simulations and theoretical analysis to the results that would be given by existing neural network guarantees under the same conditions.

Contents

List of Figures

List of Tables

Chapter 1

Context

In theoretical machine learning papers, when dealing with performance guarantees of neural networks, results mostly rely on the framework of uniform convergence. In more details, people usually start with the definition of the distribution $\mathcal{D}$ over the set $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y} \subseteq \mathbb{R}$ are domains of covariates and target variable respectively. In regression problem set $\mathcal{Y}$ is set of real numbers $\mathbb{R}$, while in binary classification problem set $\mathcal{Y}$ equals set $\{-1, 1\}$. $\mathcal{H}$ is defined to be hypotheses space where each hypothesis $h \in \mathcal{H}$ is a mapping from $\mathcal{X}$ to $\mathbb{R}$. They additionally define loss function $L : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, expected loss $L_{\mathcal{D}}(h) = E_{(x,y)\sim\mathcal{D}}[L(h(x), y)]$ and empirical loss $L_{emp,\mathcal{S}}(h) = \frac{1}{n}\sum_{(x,y)\in\mathcal{S}} L(h(x), y)$ on a dataset $\mathcal{S}$ of n datapoints, $\{(x_i, y_i) \sim \mathcal{D}, 1 \leq i \leq n\}$. In the above, $\mathbb{R}_{\geq 0}$ stands for real numbers that are greater than or equal to 0. Finally, apart from what was presented so far, people also define learning algorithm $A : \mathcal{S} \to \mathcal{H}$ which takes training set $\mathcal{S} \sim \mathcal{D}^n$ and outputs the hypothesis $h_{\mathcal{S}}$.

1.1 Generalisation guarantees

Definitions 1.1.0.1, 1.1.0.2, and 1.1.0.3 (as written below) are given in Nagarajan and Kolter (2019). These definitions are part of uniform convergence framework used to try and determine how well algorithm behaves in practice. First, generalisation error is used in order to try and determine how well learning algorithm A (as defined above) behaves in practice. Generalisation error is defined in Definition 1.1.0.1.

Definition 1.1.0.1. *Generalisation error of learning algorithm A with respect to loss L is defined to be smallest value $\epsilon_{gen}(n, \delta)$ such that $P_{\mathcal{S}\sim\mathcal{D}^n}[L_{\mathcal{D}}(h_{\mathcal{S}}) - L_{emp,\mathcal{S}}(h_{\mathcal{S}}) \leq \epsilon_{gen}(n, \delta)] \geq 1 - \delta$.*

To theoretically bound generalisation error, researchers usually resolve to **uniform convergence bound** which is defined in Definition 1.1.0.2.

Definition 1.1.0.2. *Uniform convergence bound with respect to loss L is the smallest value $\epsilon_{unif}(n, \delta)$ such that $P_{\mathcal{S}\sim\mathcal{D}^n}[\sup_{h\in\mathcal{H}} |L_{\mathcal{D}}(h) - L_{emp,\mathcal{S}}(h)| \leq \epsilon_{unif}(n, \delta)] \geq 1 - \delta$. This can be rephrased as follows: Uniform convergence bound $\epsilon_{unif}(n, \delta)$ is the smallest value for which there exists a set $\mathcal{S}_\delta$ for which $P_{\mathcal{S}\sim\mathcal{D}^n}[\mathcal{S} \in \mathcal{S}_\delta] \geq 1 - \delta$ and $\sup_{\mathcal{S}\in\mathcal{S}_\delta} \sup_{h\in\mathcal{H}} |L_{\mathcal{D}}(h) - L_{emp,\mathcal{S}}(h)| \leq \epsilon_{unif}(n, \delta)$.*

Authors of Nagarajan and Kolter (2019) go further and define algorithm dependent bound in order to try and find tighter upper bound on generalisation error. Therefore, they define

tightest algorithm-dependent uniform convergence bound which is defined in Definition 1.1.0.3.

Definition 1.1.0.3. *Tightest algorithm-dependent uniform convergence bound with respect to loss L is the smallest value $\epsilon_{unif-alg}(n,\delta)$ for which there exists a set of sample sets $\mathcal{S}_\delta$ such that $P_{\mathcal{S}\sim\mathcal{D}^n}[\mathcal{S}\in\mathcal{S}_\delta]\geq 1-\delta$ and at the same time it holds as follows: $sup_{\mathcal{S}\in\mathcal{S}_\delta} sup_{h\in\mathcal{H}_\delta}|L_{\mathcal{D}}(h)-L_{emp,\mathcal{S}}(h)|\leq\epsilon_{unif-alg}(n,\delta)$, where $\mathcal{H}_\delta=\bigcup_{\mathcal{S}\in\mathcal{S}_\delta}\{h_{\mathcal{S}}\}$ is a space of hypotheses explored by algorithm A on $\mathcal{S}_\delta$.*

In Nagarajan and Kolter (2019), authors empirically show that even such algorithm dependent uniform convergence bounds are not fine-grained enough to capture good generalisation properties of deep neural networks. Furthermore, they provably derive the same fact for linear regression model trained by the gradient descent (GD) algorithm.

1.2 Typical interpolators vs generalisation bounds

In Theisen et al. (2021), it is argued by the authors that uniform convergence framework bounds, such as those mentioned in Definitions 1.1.0.1, 1.1.0.2, and 1.1.0.3, are not fine grained enough to capture good generalisation performance observed in deep learning. They support this by saying that for complex hypothesis space $\mathcal{H}$, such as in the case of deep neural networks, worst-case estimator $h_{worst}\in\mathcal{H}$ interpolating the training data $\mathcal{S}_{train}$ may indeed have big expected loss $L_{\mathcal{D}}(h_{worst})$. At the same time, such interpolator (it interpolates training set) may be very unlikely to be selected by optimization algorithm in reality. Therefore, authors propose that it may be much better to look at the behaviour of a *typical interpolator*. In other words, this means to look into how well interpolators perform, that in reality actually might be faced.

1.3 Related results

In Theisen et al. (2021), authors analyze behaviour of *typical interpolator* by looking at the performance of random interpolators in case of binary classification problem. In their setting, hypothesis space is $\mathcal{H}=\mathcal{F}_\phi=\{f(x)=sign(w^T\phi(x)):w\in\mathbb{R}^{\mathbb{N}}\}$, where $\phi:\mathbb{R}^d\rightarrow\mathbb{R}^{\mathbb{N}}$ stands for feature mapping. This model is a special case of half-space learning model that we analyze and that is defined in Definition 1.4.0.1. Authors deal with overparameterized setting, meaning $n<d$, where n and d are the train set size and feature dimension respectively. They run the experiments on 0 vs 1 classification task on MNIST dataset with $\phi(x)=x\in\mathbb{R}^{784}$ and show that for $n=350$ samples nearly 100% of interpolators get test error less than 5%. Most importantly, they observe that overwhelming proportion of interpolators have good test set performance, but nevertheless there exist bad interpolators that perform with test error close to 1 on the test set, though extremely rare. Furthermore, they obtain consistent observations while performing same experiments on FASHION-MNIST dataset. In order to be able to vary feature dimension d of dataset, they also perform experiments on synthetic dataset generated from Gaussian mixture distribution $(x,y)\sim\mathcal{D}\sim\frac{1}{2}(\mathcal{N}_+,1)+\frac{1}{2}(\mathcal{N}_-,-1)$, where $\mathcal{N}_+\sim\mathcal{N}(\mu,\Sigma)$, and $\mathcal{N}_-\sim\mathcal{N}(-\mu,\Sigma)$. From these, they observe that as dimension d increases, test errors for random interpolators tend to concentrate around critical value ϵ^*. They also observe that the value of ϵ^* depends on the fraction between the number of training samples n and the dimension of distribution d: $\alpha=\frac{n}{d}$. They also show that ϵ^* depends on the signal to noise ratio $SNR=\sqrt{\mu^T\Sigma^{-1}\mu}$ of the underlying distribution $\mathcal{D}$. They furthermore

look into the setting with random features by having $\phi(x)$ be equal to $\sigma(Ux)$, with rows $u_1, u_2, \ldots, u_N$ of U drawn from the uniform distribution on the sphere S^{d-1} and function $\sigma(x)$ being equal to $max(x, 0)$. They apply this setting on MNIST and FASHION-MNIST dataset while having $\alpha = n/N$ being constant and observe (same as for Gaussian mixture setting above) that as the number of features N grows, that errors begin to concentrate around $\epsilon^*(\alpha)$ which further depends on difficulty of the task. Finally, they provably show some facts for the hypothesis space $\mathcal{F}_\phi$ as presented above and the setting where training and testing samples have fixed correlation in a sense that following holds: $\forall i \neq j \in \mathcal{S}_{train} \cup \mathcal{S}_{test}, y_i y_j \phi(x_i)^T \phi(x_j) = \rho$. Under this setting, they prove that the probability to correctly classify a single testing point converges to 1 with $O(\frac{1}{n})$ rate. Another thing they show is that as $n\rho \to \infty$ then $\epsilon^* = \frac{1-\rho}{n\rho}$ behaves as a critical value. In Plan and Vershynin (2013), they work on a random hyperplane tessellations problem which is strongly related to half-space learning problem. There, in Theorem 1.2 they make statement on how many random hyperplanes one needs to get δ-uniform tessellation. Loosely speaking, in our case of noiseless half-space learning model language, this is equivalent to saying how many (x, y) pairs we need in order to be able to decode signal α up to δ precision. Nevertheless, their theorem works only for cases when the number of training samples n equals $\Omega(d)$, where d is the dimension of the signal. Since the goal of our thesis is to work in the interpolating regime where $n = O(d)$, we are unable to directly apply the mentioned theorem. In their work, they further assume signal α to be sparse in order to make the number of needed training data n to be of order $O(d)$. Nevertheless, in order to stay consistent with paper Theisen et al. (2021), we are interested in a more general case where no assumptions about signal α are made.

1.4 Contribution

In this thesis, we try and extend results obtained in Theisen et al. (2021) (as explained above). In particular, we try and provably show results related to *typical interpolators* under much less artificial environment than the one in Theisen et al. (2021). There, they use fixed correlation setting (as explained above) which seems utterly artificial. Thereafter, we work with a much more general model as defined in Definition 1.4.0.1.

Definition 1.4.0.1 (Halfspace learning data model).

$$y = sign(\langle x, \alpha \rangle + \epsilon), \epsilon \sim \mathbb{N}(0, \sigma^2), x_i \sim \mathbb{N}(0, I), \alpha \in \mathcal{S}^{d-1}$$

ϵ represents noise and α is signal.

The goal of halfspace learning is to estimate α (signal vector) given the training dataset $\mathcal{S}_{train}$. We work with noiseless case of halfspace learning model ($\sigma^2 = 0$) and put our focus on overparametrized setting where dimension of signal d is greater than the number of train datapoints n and onto how well interpolating classifiers work. We prove that for size of training dataset $n = 1$, as dimension d goes to infinity, worst case interpolator behaviour converges to a behaviour much worse of the value to which typical interpolators converge. Furthermore, we present simulation results a the more general scenario when n is greater than 1. In the end, by making a connection between our model and neural network, we compare our convergence bounds to existing guarantees for neural networks.

Chapter 2

Halfspace learning model

In this chapter we work with the model presented in Definition 1.4.0.1. With $(x, y) \sim \mathcal{D}_{halfspace}$ we denote that pair (x, y) is generated as presented in Definition 1.4.0.1. For this particular model, hypothesis class is $\mathcal{H} = \{h_w(\cdot) = sign(\langle \cdot, w \rangle) : w \in \mathcal{S}^{d-1}\}$. In this chapter, because of convenience, we will use w to denote hypothesis $h_w \in \mathcal{H}$. With $\mathcal{S}_{train} = \{(x_i, y_i) : \forall i \in \{1, ..., n\}(x_i, y_i) \sim \mathcal{D}_{halfspace}\}$ we denote training set of size n. With $\mathcal{H}_{inter} = \{h_w \in \mathcal{H} : h_w(x_i) = y_i, \forall (x_i, y_i) \in \mathcal{S}_{train}\}$ we denote interpolators (set of hypothesis that interpolate the training set $\mathcal{S}_{train}$).

2.1 Loss function

Natural loss function to use in this problem is the probability of misclassification (0-1 loss). In other words, if we denote real signal with α, and estimator (of α) with w, loss function $L_D(w)$ is as follows:

$$L_D(w) = P[sign(\langle x, \alpha \rangle) \neq sign(\langle x, w \rangle)], \text{ where } x \sim \mathbb{N}(0, I) \tag{2.1.0.1}$$

As we will explain, instead of loss function (2.1.0.1) we can use L_2 distance between α and w. We will do so because of mathematical simplicity. In what follows we explain why it is justified to actually use L_2 distance. For that, we first present Theorem 2.1.0.1.

Theorem 2.1.0.1. *Consider random vector $x \sim \mathbb{N}(0, I)$. For any $u, v \in \mathcal{S}^{d-1}$, we have:*

$$P[sign(\langle x, u \rangle) \neq sign(\langle x, v \rangle)] = \frac{1}{2} - \frac{1}{\pi} sin^{-1}(\langle u, v \rangle)$$

Proof. From *Grothendieck's identity* (Lemma 3.6.6) in Vershynin (2018) it follows that:

$$E[sign(\langle x, u \rangle) sign(\langle x, v \rangle)] = \frac{2}{\pi} sin^{-1}(\langle u, v \rangle) \tag{2.1.0.2}$$

(2.1.0.2) can also be written as:

$$\begin{aligned} E[sign(\langle x, u \rangle) sign(\langle x, v \rangle)] = P(sign(\langle x, u \rangle) \geq 0, sign(\langle x, v \rangle) \geq 0) \\ + P(sign(\langle x, u \rangle) \leq 0, sign(\langle x, v \rangle) \leq 0) - P(sign(\langle x, u \rangle) \geq 0, sign(\langle x, v \rangle) \leq 0) \\ - P(sign(\langle x, u \rangle) \leq 0, sign(\langle x, v \rangle) \geq 0) \end{aligned} \tag{2.1.0.3}$$

At the same time $P(sign(\langle x,u\rangle) \neq sign(\langle x,v\rangle))$ can be written as:

$$\begin{aligned}
P(sign(\langle x,u\rangle) \neq sign(\langle x,v\rangle)) &= 1 - P(sign(\langle x,u\rangle) = sign(\langle x,v\rangle)) \\
&= 1 - P(sign(\langle x,u\rangle) \geq 0, sign(\langle x,v\rangle) \geq 0) \\
&\quad - P(sign(\langle x,u\rangle) \leq 0, sign(\langle x,v\rangle) \leq 0) \\
&\overset{\text{Law of total probability}}{=} \frac{1}{2} + \frac{P(sign(\langle x,u\rangle) \geq 0, sign(\langle x,v\rangle) \geq 0)}{2} \\
&\quad + \frac{P(sign(\langle x,u\rangle) \leq 0, sign(\langle x,v\rangle) \leq 0)}{2} \\
&\quad + \frac{P(sign(\langle x,u\rangle) \geq 0, sign(\langle x,v\rangle) \leq 0)}{2} + \frac{P(sign(\langle x,u\rangle) \leq 0, sign(\langle x,v\rangle) \geq 0)}{2} \\
&\quad - P(sign(\langle x,u\rangle) \geq 0, sign(\langle x,v\rangle) \geq 0) - P(sign(\langle x,u\rangle) \leq 0, sign(\langle x,v\rangle) \leq 0) \\
&= \frac{1}{2} - \frac{P(sign(\langle x,u\rangle) \geq 0, sign(\langle x,v\rangle) \geq 0)}{2} - \frac{P(sign(\langle x,u\rangle) \leq 0, sign(\langle x,v\rangle) \leq 0)}{2} \\
&\quad + \frac{P(sign(\langle x,u\rangle) \geq 0, sign(\langle x,v\rangle) \leq 0)}{2} + \frac{P(sign(\langle x,u\rangle) \leq 0, sign(\langle x,v\rangle) \geq 0)}{2} \\
&\overset{(2.1.0.3)}{=} \frac{1}{2}(1 - E[sign(\langle x,u\rangle)sign(\langle x,v\rangle)]) \\
&\overset{(2.1.0.2)}{=} \frac{1}{2} - \frac{1}{\pi}sin^{-1}(\langle u,v\rangle)
\end{aligned}$$

□

Next, we denote that it holds that $||\alpha||_2 = ||w||_2 = 1$, and that inner product of two vectors u and v can be rewritten as $\langle u,v\rangle = \frac{||u||_2^2+||v||_2^2-||u-v||_2^2}{2}$. Using Theorem 2.1.0.1 and all of the above, we get that loss function (2.1.0.1) can be rewritten as a function of L_2 distance between signal α and estimator w as follows:

$$L_D(w) = \frac{1}{2} - \frac{1}{\pi}sin^{-1}(1 - \frac{1}{2}||\alpha - w||_2^2) \tag{2.1.0.4}$$

Therefore, from (2.1.0.4) it follows that there is a deterministic connection between the 0-1 loss and L_2 distance. Next, we provide further arguments for why it is actually justified (deterministic connection is not enough) to exchange 0-1 loss from (2.1.0.1) with simpler Euclidean distance loss $||\alpha - w||_2$. While doing so, we will interchangebly be using $d(a,b)$ and $||a-b||_2$ for the Euclidean distance between points $a,b \in \mathcal{S}^{d-1}$. Additionally, we will use notation $g(a,b)$ for the geodesic distance between points $a,b \in \mathcal{S}^{d-1}$ on the sphere. Next, we present a theorem making a connection between Euclidean and geodesic distance on a sphere.

Theorem 2.1.0.2. *Let $a,b \in \mathcal{S}^{d-1}$. Then $d(a,b) = 2\sin\frac{g(a,b)}{2}$.*

Proof. Proof closely follows notations in Figure 2.1. From simple arythmetic expression on right-angled triangle (containing coordinate system origin, red line, and line connecting origin with point b), we get that:

$$\frac{\frac{d(a,b)}{2}}{1} = sin(\frac{\phi}{2}) \overset{reorganizing}{\Rightarrow} d(a,b) = 2sin(\frac{\phi}{2}) \tag{2.1.0.5}$$

At the same time, for $g(a, b)$ it holds that:

$$g(a,b) = \underbrace{1}_{\text{radius of the circle}} \phi = \phi \tag{2.1.0.6}$$

Inserting (2.1.0.6) in (2.1.0.5) we get expression relating Euclidean and geodesic distance as follows:

$$d(a,b) = 2sin(\frac{g(a,b)}{2}) \tag{2.1.0.7}$$

$\square$

Corollary 2.1.0.3. *Let $a, b \in \mathcal{S}^{d-1}$. Then $||a - b||_2 \leq g(a, b) \leq \frac{\pi}{2}||a - b||_2$.*

Proof. From Theorem 2.1.0.2 we have a deterministic connection between geodesic and Euclidean distance as $d(a, b) = 2\sin(\frac{g(a,b)}{2})$. Next, function $f(x) = \frac{x}{2\sin(\frac{x}{2})}$ is monotonically increasing for $x \in [0, \pi]$. This is easily seen using simple calculus on a function f as follows:

$$f'(x) = \frac{2\sin(\frac{x}{2}) - x\cos(\frac{x}{2})}{\underbrace{4\sin^2(\frac{x}{2})}_{\geq 0}} \tag{2.1.0.8}$$

Next, we denote numerator of (2.1.0.8) as $g(x) = 2\sin(\frac{x}{2}) - x\cos(\frac{x}{2})$. For function g, following holds.

$$g'(x) = \frac{x}{2}\sin(\frac{x}{2}) \geq 0, \text{ for } x \in [0, \pi] \tag{2.1.0.9}$$
$$g(0) = 0$$

From (2.1.0.9), it directly follows that the numerator of $f'(x)$, which we denote as $g(x)$, is greater or equal to 0 for $x \in [0, \pi]$. From this and (2.1.0.8), it further follows that $f(x)$ is monotonically increasing for $x \in [0, \pi]$. For $x = \pi$, it holds that $f(x) = \frac{\pi}{2}$. The numerator of f can be represented as a geodesic distance, while the denominator of f can be represented as an Euclidean distance. From all that and Theorem 2.1.0.2, it follows that $g(a, b)$ is lower than or equal to $\frac{\pi}{2}||a - b||_2$. Lower bound $||a - b||_2 \leq g(a, b)$ trivially follows from the fact that the geodesic distance is an arc above the segment whose length corresponds to Euclidean distance. $\square$

Next, in Theorem 2.1.0.4, we make a connection between geodesic distance and 0-1 loss. After that, by using the relation between Euclidean and geodesic distance from Theorem 2.1.0.2, we make a pathway to the reason behind why it is justified to use Euclidean distance instead of 0-1 loss as a loss function.

Theorem 2.1.0.4. *Let $a, b \in \mathcal{S}^{d-1}$. For $g \sim \mathbb{N}(0, I)$, it holds that $P(sign(\langle g, a\rangle) \neq sign(\langle g, b\rangle)) = \frac{g(a,b)}{\pi}$.*

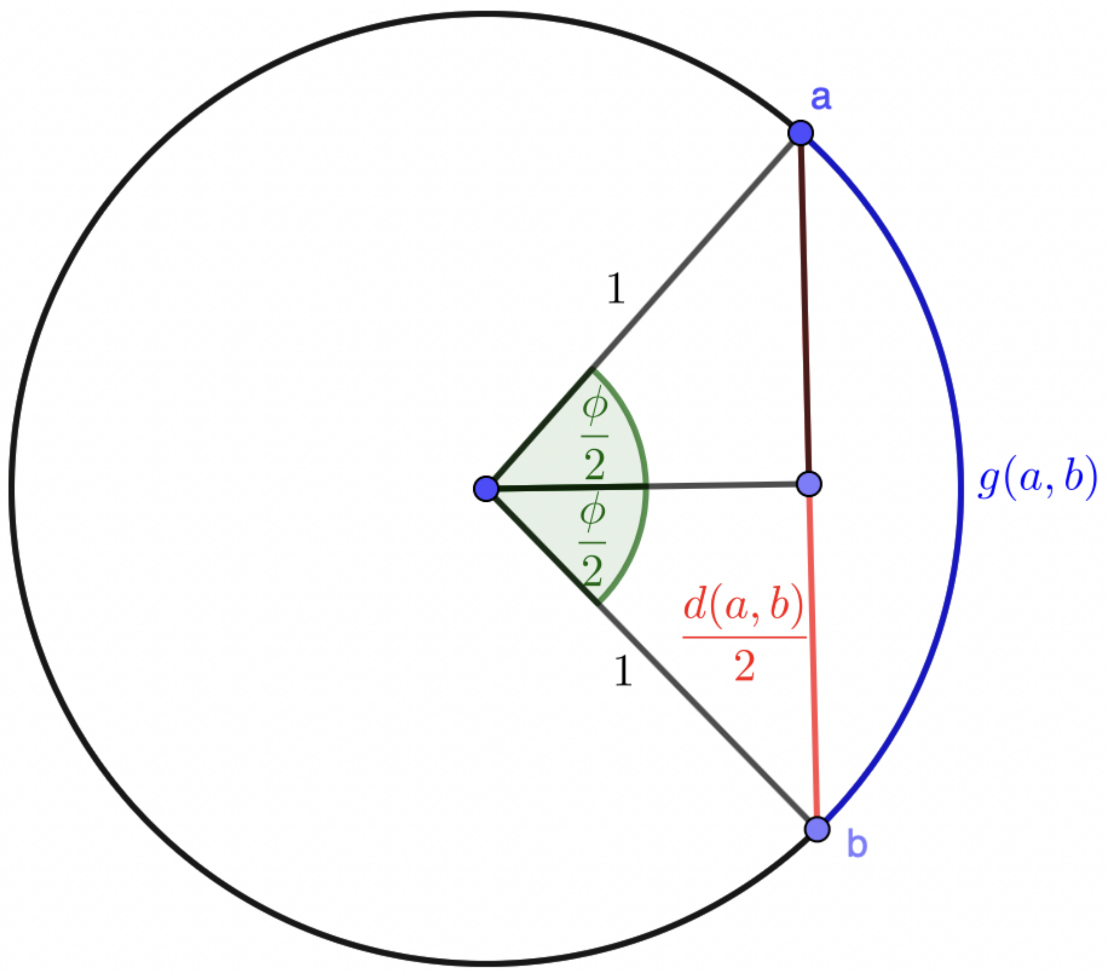


Figure 2.1: Figure shows a 2-dimensional plane spanned by vectors $a \in \mathcal{S}^{d-1}$ and $b \in \mathcal{S}^{d-1}$. The origin of the coordinate system is in the center of the circle. As denoted, the radius of the circle is 1. The angle between points a and b, with a center in the origin of the coordinate system, is denoted with ϕ. The line presenting half of the euclidean distance $||a-b||_2$ is coloured in red and denoted with d(a,b)/2. Geodesic distance between points a and b is coloured in blue and denoted with $g(a,b)$.

Proof. Project g onto the span of a, b (called $span(a,b)$) and call it $proj_{a,b}(g)$. Because of the rotational invariance of distribution of g, $\frac{proj_{a,b}(g)}{||proj_{a,b}(g)||_2}$ is uniformly distributed on $\mathcal{S}^1$ that lies on the $span(a,b)$. Furthermore, by using the fact that g can be rewritten as $g = proj_{a,b}(g) + proj_{a,b}^T(g)$, where $proj_{a,b}^T(g)$ is the projection of g on a complement of the space $span(a,b)$, it follows trivially that the events $\{sign(\langle g,a\rangle) \neq sign(\langle g,b\rangle)\}$ and $\{sign(\langle \frac{proj_{a,b}(g)}{||proj_{a,b}(g)||_2}, a\rangle) \neq sign(\langle \frac{proj_{a,b}(g)}{||proj_{a,b}(g)||_2}, b\rangle)\}$ are the same. Therefore, $P(sign(\langle g,a\rangle) \neq sign(\langle g,b\rangle))$ is equal to the red portion of the circle in Figure 2.2 as follows:

$$P(sign(\langle g,a\rangle) \neq sign(\langle g,b\rangle)) = P(\text{red area}) = \frac{2\phi}{2\pi} = \frac{\phi}{\pi} \overset{(2.1.0.6)}{=} \frac{g(a,b)}{\pi} \tag{2.1.0.10}$$

□

From Theorem 2.1.0.4 (where we make a connection between 0-1 loss and geodesic distance) and Corollary 2.1.0.3 (bounds on geodesic distance using Euclidean distance), we can finally make a bound on 0-1 loss using the Euclidean distance between estimator w and real signal α as stated in the next theorem (just replace a and b with α and w).

Theorem 2.1.0.5. *Let $a, b \in \mathcal{S}^{d-1}$. For $g \sim \mathbb{N}(0, I)$, it holds that $||a-b||_2 \leq \pi P(sign(\langle g,a\rangle) \neq sign(\langle g,b\rangle)) \leq \frac{\pi}{2}||a-b||_2$.*

Proof. This directly follows from Theorem 2.1.0.4, and Corollary 2.1.0.3. □

Finally, because of deterministic connection between 0-1 loss and Euclidean distance from (2.1.0.4), the fact that the function $f(x) = \frac{1}{2} - \frac{1}{\pi}sin^{-1}(1 - \frac{1}{2}x^2)$ is monotonically increasing for $x \in [0,2]$ and bounds on 0-1 loss using the Euclidean distance from Theorem 2.1.0.5, it follows that we can replace 0-1 loss with Euclidean distance of estimator w from signal α, and that it will truthfully represent the goodness of the estimator w. Additionally, on Figure 2.3, we give plot of a function $f(x) = \frac{1}{2} - \frac{1}{\pi}sin^{-1}(1 - \frac{1}{2}x^2)$ for $x \in [0,2]$. There, we can see 0-1 loss as a function of L_2 distance and be assured that interchangeability of 0-1 and L_w distance loss is allowed.

2.2 Assumptions

In most of what follows in this chapter, we rigurously analyse behaviour of quantities $S_{dist} = \sup_{w: w \in \mathcal{H}_{inter}} ||\alpha - w||_2$ and $E_{dist} = \mathbb{E}[||\alpha - w||_2 | w \in \mathcal{H}_{inter}]$ for the noiseless case of data generating model as presented in Definition 1.4.0.1. But first, we lay out all the assumptions we make while doing so.

Remark. Without loss of generality we can assume signal vector $\alpha = (1, 0, \cdots, 0)$.

Proof. Suppose real $\alpha \neq (1, 0, \cdots, 0)$. Then we can rotate our coordinate system so that α' (coordinates of α in a new coordinate system) is $(1, 0, \cdots, 0)$. In a rotated coordinate system, because of rotational invariance of distribution of $x \sim \mathbb{N}(0, I)$, and since we are working with a noiseless case of the model in Definition 1.4.0.1 (meaning that $y = sign(\langle x, \alpha\rangle)$), nothing changes regarding the distribution of pairs (x, y). Since the interpolator set $\mathcal{H}_{inter}$ depends only on iid. pairs of (x, y) throughout the train set $\mathcal{S}_{train}$, it follows that nothing changes regarding the distribution of interpolator set $\mathcal{H}_{inter}$. □

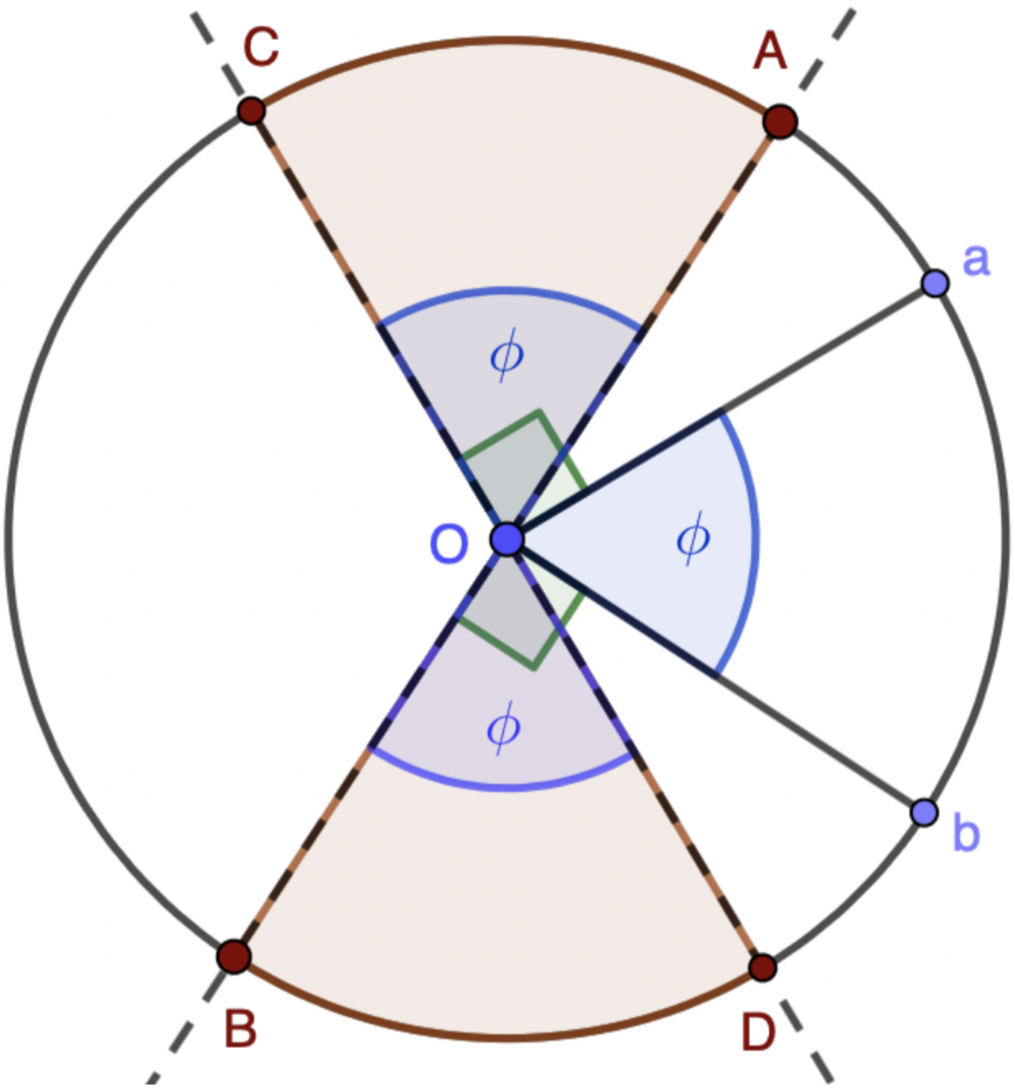


Figure 2.2: Figure shows a 2-dimensional plane spanned by vectors u and v. In particular, we show intersection of the plane with $\mathcal{S}^{d-1}$ sphere with a black circle. The area from which we pick element x, in which holds $sign(\langle x, u\rangle) \neq sign(\langle x, v\rangle)$, is shaded with a red color.

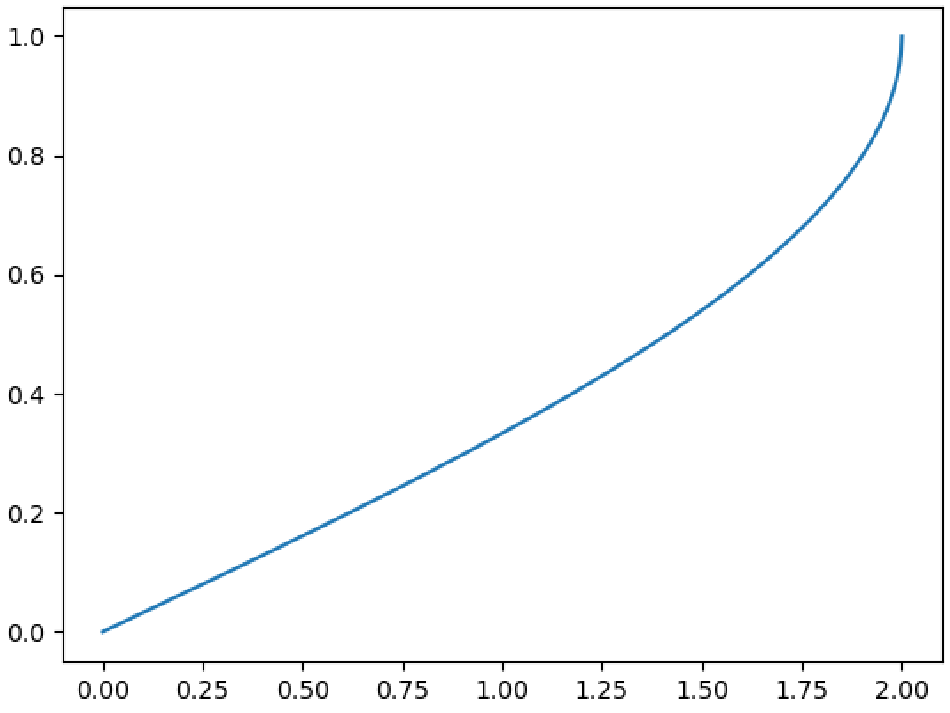


Figure 2.3: A graph showing function $f(x) = \frac{1}{2} - \frac{1}{\pi} sin^{-1}(1 - \frac{1}{2}x^2)$ which relates 0-1 loss to Euclidean distance as presented in (2.1.0.4).

Remark. The effect of pairs $(x, 1)$ and $(-x, -1)$ are the same on the geometry of the set $\mathcal{H}_{inter}$.

Proof. Suppose that $\mathcal{S}_{train}$ is such that for some $i \in \{1, 2, \cdots, n\}$ $(x_i, y_i) = (x_i, -1)$. From the definition of $\mathcal{H}_{inter}$, it follows that $\forall h_w \in \mathcal{H}_{inter}$, $h_w(x_i) = sign(\langle x_i, w\rangle) = y_i = -1$. This holds if and only if $h_w(-x_i) = sign(\langle -x_i, w\rangle) = -sign(\langle x_i, w\rangle) = -y_i = 1$. Therefore, if we replace sample $(x_i, y_i) = (x_i, -1)$ with $(-x_i, 1)$, we do not change the set $\mathcal{H}_{inter}$. □

Remark. Without loss of generality we can assume $\mathcal{S}_{train}$ is such that for all $i \in \{1, 2, \cdots, n\}$: $(x_i, y_i) = (x_i, 1)$.

Proof. From Remark 2.2, it follows that we can exchange each pair $(x_i, y_i) \in \mathcal{S}_{train}$ for which $y_i = -1$ with $(-x_i, 1)$ and not change the geometry of the set $\mathcal{H}_{inter}$. Therefore, since we do not change the geometry of the set $\mathcal{H}_{inter}$, we can do the switching operation and at the same time not change the distribution of the set $\mathcal{H}_{inter}$. □

Remark. Without loss of generality we can assume that samples (x_i, y_i) are such that $y_i = 1$ and x_i is sampled from the uniform distribution on a d-dimensional half-sphere for which it holds that $\langle x_i, \alpha\rangle \geq 0$.

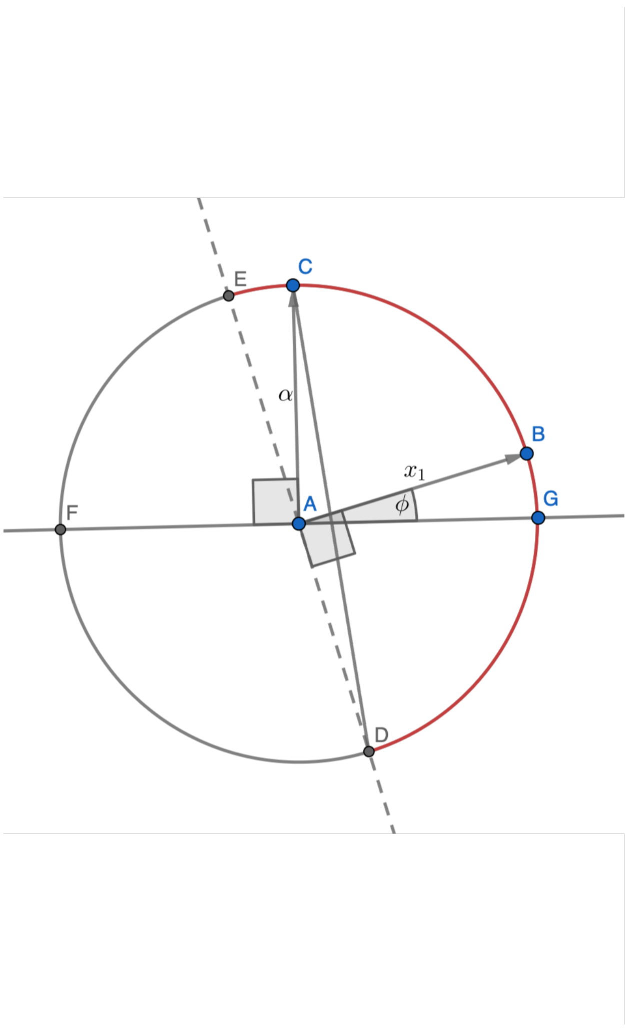
Proof. This follows directly from equivalency of the uniform distribution on a d-dimensional sphere and a normalized d-dimensional multivariate normal distribution $\mathbb{N}(0, I_{d\times d})$, and Remarks 2.2 and 2.2. □

Using the remarks above, we finally start to analyse quantities E_{dist} (typical interpolator) and S_{dist} (worst case interpolator). In particular, we deal with the case when $|\mathcal{S}_{train}| = n = 1$. We have $\mathcal{S}_{train} = \{(x_1, y_1)\}$. The idea is to bound the position of x_1 with a high probability event and show that under this high probability scenario, quantities E_{dist} and S_{dist} behave in a particular way.

2.3 Analysis of worst case interpolator

Lemma 2.3.0.1. *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, positive constant $\epsilon > 0$ and dimension $d > 2$, with probability $p \geq 1 - \frac{2}{\epsilon\sqrt{d-2}} e^{-\frac{d-2}{2}\epsilon^2}$, it holds that $l \geq \frac{\epsilon}{\sqrt{\frac{1-\sqrt{1-\epsilon^2}}{2}}}$. l and other quantities of interest are denoted in the Figure 2.4.*

Proof. From section 2.2.5 in Hopcroft and Kannan (Hopcroft and Kannan), we have that with probability $p \geq 1 - \frac{2}{\epsilon\sqrt{d-2}} e^{-\frac{d-2}{2}\epsilon^2}$, $x_1 = (x_{1,1}, x_{1,2}, \cdots x_{1,d})$ is such that $0 \leq x_{1,1} \leq \epsilon$. We denote this event as $F(d, \epsilon)$. Next, we lower bound the l under the event $F(d, \epsilon)$. For that, we analyse a 2-dimensional hyperplane spanned by vectors x_1 and α. Calculations closely follow notations in Figure 2.4.



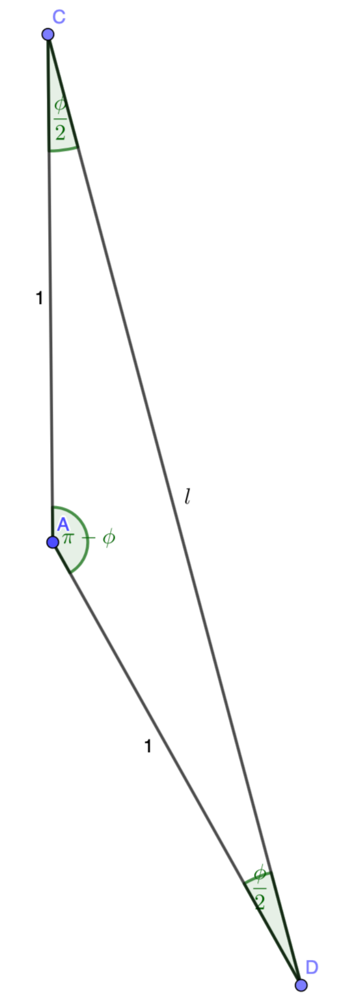


Figure 2.4: **Left:** A 2-dimensional hyperplane spanned by vectors x_1 and α. The circle shows an intersection of a hyperplane with the a d-dimensional sphere centered at point A. The part of the circle which also belongs to the interpolator set $\mathcal{H}_{inter}$ is shown with a red color. ϕ is the angle between the vector x_1 and the line perpendicular to vector α. **Right:** A zoomed in triangle $\triangle ADC$ from the left part of the figure. Here, it is also taken into account that both the length of vector α and the radius of the hypersphere are 1.

$$\frac{sin(\pi-\phi)}{l} = \frac{sin(\frac{\phi}{2})}{1} \text{ (law of sines)}$$
$$\overset{sin(\pi-\phi)=sin(\phi)}{\Rightarrow} \frac{sin(\phi)}{l} = sin(\frac{\phi}{2})$$
$$\overset{sin(\phi)=x_{1,1},\ sin(\frac{\phi}{2})=\sqrt{\frac{1-cos(\phi)}{2}}}{\Rightarrow} \frac{x_{1,1}}{l} = \sqrt{\frac{1-cos(\phi)}{2}}$$
$$\overset{cos(\phi)=\sqrt{1-x_{1,1}^2}}{\Rightarrow} \frac{x_{1,1}}{l} = \sqrt{\frac{1-\sqrt{1-x_{1,1}^2}}{2}}$$
$$\Rightarrow l = \frac{x_{1,1}}{\sqrt{\frac{1-\sqrt{1-x_{1,1}^2}}{2}}}$$

Since $x_{1,1} \leq \epsilon$, and because function $f(x) = \frac{x}{\sqrt{\frac{1-\sqrt{1-x^2}}{2}}}$ is monotonically decreasing for x, it follows that $l \geq \frac{\epsilon}{\sqrt{\frac{1-\sqrt{1-\epsilon^2}}{2}}}$. □

Theorem 2.3.0.2. *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, positive constant $\epsilon > 0$ and dimension $d > 2$, with probability $p \geq 1 - \frac{2}{\epsilon\sqrt{d-2}}e^{-\frac{d-2}{2}\epsilon^2}$, it holds that $S_{dist} \geq \frac{\epsilon}{\sqrt{\frac{1-\sqrt{1-\epsilon^2}}{2}}}$.*

Proof. This follows directly from Lemma (2.3.0.1) and the fact that $S_{dist} \geq l$. □

Corollary 2.3.0.3. *Let $\mathcal{S}_{train} = \{(x_1, y_1)\}$. $S_{dist} \to 2$ a.s. as dimension $d \to \infty$.*

Proof. By setting $\epsilon = \frac{c}{\sqrt[4]{d-2}}$ (for some positive constant c) in Theorem 2.3.0.2, we get that for event $F(d, \epsilon = \frac{c}{\sqrt[4]{d-2}})$ (as defined in the proof of Lemma 2.3.0.1) it holds:

$$P(F(d, \epsilon = \frac{c}{\sqrt[4]{d-2}})) \geq 1 - \frac{2}{c\sqrt[4]{d-2}}e^{-\frac{\sqrt{d-2}}{2}c^2}$$

For $d \to \infty$ it follows:

$$\lim_{d\to\infty} P(F(d, \epsilon = \frac{c}{\sqrt[4]{d-2}})) \geq \lim_{d\to\infty}(1 - \frac{2}{c\sqrt[4]{d-2}}e^{-\frac{\sqrt{d-2}}{2}c^2}) = 1$$

At the same time it holds that $S_{dist} \geq \frac{c}{\sqrt[4]{d-2}\sqrt{\frac{1-\sqrt{1-\frac{c^2}{\sqrt{d-2}}}}{2}}}$, and as $\lim_{d\to\infty}$ we have:

$$S_{dist} \geq \lim_{d\to\infty} \frac{c}{\sqrt[4]{d-2}\sqrt{\frac{1-\sqrt{1-\frac{c^2}{\sqrt{d-2}}}}{2}}} = 2$$

From what is written above, and trivial upper bound $S_{dist} \leq 2$, it follows that:

$$\lim_{d \to \infty} S_{dist} = 2 \text{ a.s.} \tag{2.3.0.1}$$

□

2.4 Analysis of typical interpolator

Lemma 2.4.0.1. *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, positive constant $\epsilon > 0$, and dimension d, the probability that a randomly sampled $w = (w_1, w_2, \cdots, w_d) \sim \mathbb{P}(\cdot|\mathcal{H}_{inter})$ is such that $-\epsilon \leq w_1 \leq \epsilon$ is equal to $P(z \in W_\epsilon)$, where z is uniformly distributed on the halfsphere $\langle z, \alpha \rangle \geq 0$ and $W_\epsilon = \{w : 0 \leq w_1 \leq \epsilon, ||w||_2 = 1\}$. In the above, $\mathbb{P}(\cdot|\mathcal{H}_{inter})$ is the uniform probability measure on the halfsphere defined by $\{w : \langle w, x1 \rangle \geq 0\}$.*

Proof. In what follows, we work with uniform probability measures on a unit sphere, and use the notation $\mu(A)$ for the measure of a set $A \subseteq H$ under the uniform probability measure on the unit sphere H. Next, we show that uniform probability measure of the a.s. nonempty set $W_\epsilon \cap \mathcal{H}^c_{inter}$ and the set $W_{-\epsilon} \cap \mathcal{H}_{inter}$ are the same by creating an isometry mapping between the two. Let $\mathcal{T} : \mathcal{R}^d \to \mathcal{R}^d$ be defined as a mapping from the vector $v \in \mathcal{R}^d$ into the vector $-v \in \mathcal{R}^d$. First, we show that mapping $\mathcal{T}$ indeed maps the set $W_\epsilon \cap \mathcal{H}^c_{inter}$ into the set $W_{-\epsilon} \cap \mathcal{H}_{inter}$. Take some $w = (w_1, w_2, \cdots, w_d) \in W_\epsilon \cap \mathcal{H}^c_{inter}$. For any such w, it holds that $\langle w, x_1 \rangle \leq 0$. For $w' = \mathcal{T}(w) = -w$, it holds that $\langle w', x_1 \rangle = \langle -w, x_1 \rangle = -\underbrace{\langle w, x_1 \rangle}_{\leq 0} \geq 0$. This means that $w' \in \mathcal{H}_{inter}$. Additionally, from construction, it follows that $-\epsilon \leq w'_1 = -w_1 \leq 0$, and that $||w'||_2 = ||-w||_2 = ||w||_2 = 1$. This means that $w' \in W_{-\epsilon}$. Therefore, it also holds that $w' \in W_{-\epsilon} \cap \mathcal{H}_{inter}$. From construction itself, it is trivial to see that the mapping presented is a bijection. Next, we argue that mapping $\mathcal{T}$ is an isometry. Mapping $\mathcal{T}$ corresponds to the matrix $-\mathbb{I}$ (for the standard basis). Since $-\mathbb{I}(-\mathbb{I}) = \mathbb{I}$, $\mathcal{T}$ is in the orthogonal group $\mathcal{O}(d)$. Therefore, mapping $\mathcal{T}$ is indeed an isometry. Using this and the fact that uniform distribution is invariant under the isometry mappings, it follows that $\mu(W_\epsilon \cap \mathcal{H}^c_{inter}) = \mu(\mathcal{T}(W_\epsilon \cap \mathcal{H}^c_{inter})) = \mu(W_{-\epsilon} \cap \mathcal{H}_{inter})$. From this, it further follows that:

$$\begin{aligned}
&\mu((W_\epsilon \cup W_{-\epsilon}) \cap \mathcal{H}_{inter}) \\
&\overset{W_\epsilon \cap W_{-\epsilon} = \emptyset}{=} \mu(W_\epsilon \cap \mathcal{H}_{inter}) + \mu(W_{-\epsilon} \cap \mathcal{H}_{inter}) \\
&= \mu(W_\epsilon \cap \mathcal{H}_{inter}) + \mu(W_\epsilon \cap \mathcal{H}^c_{inter}) = \mu(W_\epsilon)
\end{aligned} \tag{2.4.0.1}$$

From (2.4.0.1), and the fact that $W_\epsilon \subseteq \{w : \langle w, \alpha \rangle \geq 0\}$ it follows that $\mu((W_\epsilon \cup W_{-\epsilon}) \cap \mathcal{H}_{inter}) = \mu(W_\epsilon \cap \{w : \langle w, \alpha \rangle \geq 0\})$. Now, both sets $\mathcal{H}_{inter}$ and $\{w : \langle w, \alpha \rangle \geq 0\}$ correspond to halfspheres and have the same uniform probability measure (again, it is possible to create an isometric mapping from one set to another in order to prove that their uniform probability measures are same). From this, we have that $\frac{\mu((W_\epsilon \cup W_{-\epsilon}) \cap \mathcal{H}_{inter})}{\mu(\mathcal{H}_{inter})} = \frac{\mu(W_\epsilon \cap \{w : \langle w, \alpha \rangle \geq 0\})}{\mu(\{w : \langle w, \alpha \rangle \geq 0\})}$. This is exactly as saying that $\mu(W_\epsilon \cup W_{-\epsilon} | \mathcal{H}_{inter}) = \mu(W_\epsilon | \{w : \langle w, \alpha \rangle \geq 0\})$, which is exactly what we needed to prove. □

Theorem 2.4.0.2. *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, positive constant $\epsilon > 0$, and dimension $d > 2$, with probability $p \geq 1 - \frac{2}{\epsilon\sqrt{d-2}}e^{-\frac{d-2}{2}\epsilon^2}$, it holds that $w = (w_1, w_2, \cdots, w_d) \sim \mathbb{P}(\cdot|\mathcal{H}_{inter})$ is such that $-\epsilon \leq w_1 \leq \epsilon$. We denote this event as $F_E(d, \epsilon)$.*

Proof. This follows directly from Theorem 2.4.0.1 and the argument from section 2.2.5 in Hopcroft and Kannan (Hopcroft and Kannan). □

Next, using corollary 2.4.0.2 presented above, we analyse quantity E_{dist}.

$$\begin{aligned} E_{dist} &= \mathbb{E}[||\alpha - w||_2 | w \in \mathcal{H}_{inter}] \\ &= \int_{\mathcal{H}_{inter}} ||\alpha - w||_2 P(dw|\mathcal{H}_{inter}) \\ &= \int_{F_E(d,\epsilon)} ||\alpha - w||_2 P(dw|\mathcal{H}_{inter}) + \int_{F_E(d,\epsilon)^C} ||\alpha - w||_2 P(dw|\mathcal{H}_{inter}) \end{aligned} \tag{2.4.0.2}$$

Let $z = ||\alpha - w||_2$. It holds that $\sqrt{2}\sqrt{1-\epsilon} \leq z \leq \sqrt{2}\sqrt{1+\epsilon}$, when w is such that $-\epsilon \leq w_1 \leq \epsilon$ (event $F_E(d, \epsilon)$ is active). We depict and explain that in Figure 2.5. On the other hand, under the event $F_E(d, \epsilon)^C$, we have trivial bound $0 \leq z \leq 2$.

Theorem 2.4.0.3 (Lower bound on E_{dist})**.** *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, positive constant $c > 0$, and dimension $d > 2$, it holds that $\sqrt{2}\sqrt{1 - \frac{c}{\sqrt[4]{d-2}}}(1 - \frac{2}{c\sqrt[4]{d-2}}e^{-\frac{\sqrt[2]{d-2}c^2}{2}}) \leq E_{dist}$.*

Proof. From (2.4.0.2) and by using the lower bounds for z on events $F_E(d, \epsilon)$ and $F_E(d, \epsilon)^C$ we have a **lower bound** on E_{dist} as follows:

$$\begin{aligned} &\sqrt{2}\sqrt{1-\epsilon}P(F_E(d,\epsilon)) \leq E_{dist} \\ \overset{P(F_E(d,\epsilon)) \geq 1 - \frac{4}{\epsilon\sqrt{d-2}}e^{-\frac{d-2}{2}\epsilon^2}}{\Rightarrow} \quad &\sqrt{2}\sqrt{1-\epsilon}(1 - \frac{4}{\epsilon\sqrt{d-2}}e^{-\frac{d-2}{2}\epsilon^2}) \leq E_{dist} \end{aligned} \tag{2.4.0.3}$$

For $\epsilon = \frac{c}{\sqrt[4]{d-2}}$, where $c \in \mathbb{R}^+$ from (2.4.0.3) it follows that:

$$\sqrt{2}\sqrt{1 - \frac{c}{\sqrt[4]{d-2}}}(1 - \frac{4}{c\sqrt[4]{d-2}}e^{-\frac{\sqrt[2]{d-2}c^2}{2}}) \leq E_{dist}$$

□

Corollary 2.4.0.4. *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, and as dimension $d \to \infty$, then following lower bound holds for E_{dist}: $\sqrt{2} \leq E_{dist}$.*

Proof. This follows directly from Theorem 2.4.0.3 and by letting $d \to \infty$. □

Theorem 2.4.0.5 (Upper bound on E_{dist})**.** *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, positive constant $c > 0$, and dimension $d > 2$, it holds that $E_{dist} \leq \sqrt{2} + 2\frac{4}{c\sqrt[4]{d-2}}e^{-\frac{\sqrt[2]{d-2}c^2}{2}}$.*

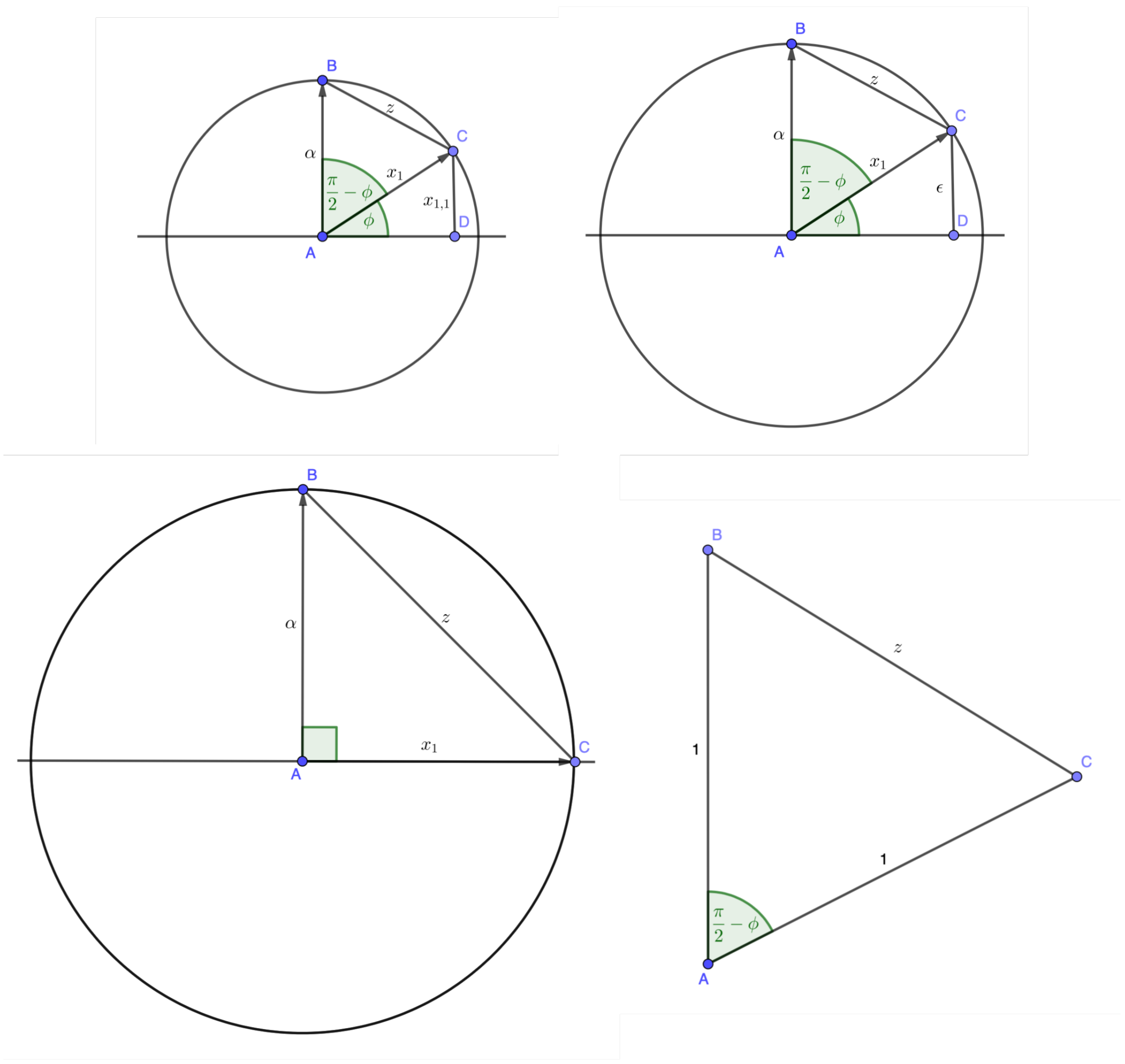


Figure 2.5: Images shown are used to depict the geometry of quantity z under the event $F_E(d, \epsilon)$. On the first 3 images, a 2-dimensional hyperplane spanned by vectors x_1 and α is shown along with z. The first image represents a general case geometry of the problem, under the event $F_E(d, \epsilon)$. The second and third images show edge cases for z when $x_{1,1} = \epsilon$ and $x_{1,1} = 0$, respectively. The fourth image shows a zoomed in triangle $\triangle ABC$ from the first 3 images (with appropriate lengths and angle). Using this triangle, it is easy to calculate z using the cosine law as follows: $z = \sqrt{1 + 1 - 2cos(\frac{\pi}{2} - \phi)} = \sqrt{2}\sqrt{1 - cos(\frac{\pi}{2} - \phi)}$. Using the fact that $cos(\frac{\pi}{2} - \phi) = sin(\phi)$ and $sin(\phi) = x_{1,1}$, it follows that: $z = \sqrt{2}\sqrt{1 - x_{1,1}}$. From this and the fact that $-\epsilon \leq x_{1,1} \leq \epsilon$ we get the bounds on z: $\sqrt{2}\sqrt{1-\epsilon} \leq z \leq \sqrt{2}\sqrt{1+\epsilon}$.

Proof. From (2.4.0.2) and by using the upper bounds for z on events $F_E(d,\epsilon)$ and $F_E(d,\epsilon)^C$, we have an **upper bound** on E_{dist} as follows:

$$E_{dist} \leq \sqrt{2}\sqrt{1+\epsilon}P(F_E(d,\epsilon)) + 2P(F_E(d,\epsilon)^C)$$

$$\overset{P(F_E(d,\epsilon))\leq 1, P(F_E(d,\epsilon)^C)\leq \frac{2}{\epsilon\sqrt{d-2}}e^{-\frac{d-2}{2}\epsilon^2}}{\Rightarrow} \quad E_{dist} \leq \sqrt{2}\sqrt{1+\epsilon} + 2\frac{2}{\epsilon\sqrt{d-2}}e^{-\frac{d-2}{2}\epsilon^2} \tag{2.4.0.4}$$

Again, by taking $\epsilon = \frac{c}{\sqrt[4]{d-2}}$ for $c \in \mathbb{R}^+$, from (2.4.0.4), it follows:

$$E_{dist} \leq \sqrt{2}\sqrt{1+\frac{c}{\sqrt[4]{d-2}}} + 2\frac{2}{c\sqrt[4]{d-2}}e^{-\frac{\sqrt{d-2}c^2}{2}}$$

□

Corollary 2.4.0.6. *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, and as dimension $d \to \infty$, then following upper bound holds for E_{dist}: $E_{dist} \leq \sqrt{2}$.*

Proof. This follows directly from Theorem 2.4.0.5 and by letting $d \to \infty$. □

Corollary 2.4.0.7. *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, as dimension $d \to \infty$, then $E_{dist} \to \sqrt{2}$.*

Proof. This follows directly from Corollary 2.4.0.4 and Corollary 2.4.0.6. □

Next, except for just proving that as dimension $d \to \infty$ quantity E_{dist} goes to $\sqrt{2}$ a.s. (shown in Corrolary 2.4.0.7), we show a stronger statement: that a.s. all interpolators $w \in \mathcal{H}_{inter}$ converge to having L_2 distance equal to $\sqrt{2}$ from the signal α. We make that rigorous in the next corollary.

Corollary 2.4.0.8. *For $\mathcal{S}_{train} = \{(x_1, y_1)\}$, as dimension $d \to \infty$, then a.s. all interpolators $w \in \mathcal{H}_{inter}$ converge to having L_2 distance from signal α equal to $\sqrt{2}$.*

Proof. Inserting $\epsilon = \frac{c}{\sqrt[4]{d-2}}$ into Theorem 2.4.0.2 gives us that with $p \geq 1 - \frac{2}{c\sqrt[4]{d-2}}e^{-\frac{\sqrt{d-2}}{2}c^2}$ we have that interpolator $w = (w_1, w_2, \cdots, w_d)$ is such that $-\frac{c}{\sqrt[4]{d-2}} \leq w_1 \leq \frac{c}{\sqrt[4]{d-2}}$. Letting d go to infinity we get that a.s. all interpolators w are such that $w_1 = 0$. By using the relation that $||w-\alpha||_2^2 = ||w||_2^2 + ||\alpha||_2^2 + 2\langle w, \alpha\rangle$, and the facts that $||\alpha||_2 = ||w||_2 = 1$ and $\langle w, \alpha\rangle = w_1 = 0$ a.s. (here we used the fact that $\alpha = (1, 0, \cdots, 0)$), we get that $||w-\alpha||_2 = \sqrt{2}$ a.s. □

2.5 Worst case interpolator vs typical interpolator

From Corollary 2.3.0.3 and Corollary 2.4.0.7, it follows that for the halfspace learning model, when $n = 1$ and $d \to \infty$, random interpolator works much better than the worst case interpolator. We denote this by saying that *typical interpolator performs much better than the worst case interpolator.* It is beneficial, other than in words of L_2 distance, to actually state how well does a typical classifier classsify when compared to a worst case classifier. For that reason, we state the next theorem.

Theorem 2.5.0.1. *For $|\mathcal{S}_{train}| = n = 1$, and for $d \to \infty$, worst case interpolator w_{worst} makes a classification error of* 1 *while a.s. all interpolators $w_{typical}$ make a classification error of $\frac{1}{2}$. (Note: classification error is the same as 0-1 loss).*

Proof. This follows directly from the deterministic relationship between 0-1 loss and L_2 distance as stated in (2.1.0.4). For the worst case interpolator, we insert the result from Corollary 2.3.0.3, while for typical interpolators we insert the result from Corollary 2.4.0.7. □

Next, we point out that our halfspace learning model can be seen as a simple 1-layer neural network with a *sign* activation function. Details regarding this statement are explained in Section 3.1. From the stated equivalence between neural networks and halfspace learning model, and by using Theorem 2.5.0.1, it follows that the uniform convergence framework might indeed be completely inapropriate to be used when dealing with performance guarantees for neural networks. This follows from the fact that even in the simple neural network case as the halfspace learning model, for case when $|\mathcal{S}_{train}| = n = 1$, worst case interpolator behaves in the worst way possible (when halfspace learning hypothesis space is taken into consideration). In the next section, we make simulations in order to explore the behaviour of the halfspace learning model for a more general n greater than 1. In particular, we are interested in model's behaviour for fixed $\zeta = \frac{n}{d}$. There, we obtain results which further support our statement that even simple neural networks are not suited to be used together with uniform convergence framework. At the same time, by taking (for example) 99^{th} percentile of a typical interpolator 0-1 loss, we would get bound that much better reflects what will happen in reality.

2.6 Simulations

In this section, we perform simulations on iid. data obtained from the halfspace learning model presented in Definition 1.4.0.1. Simulations are done for various values of $\zeta = \frac{n}{d} \in (0, 1]$, d, and n in order to see how typical interpolators behave when compared to the worst case interpolator for varying values of ζ, d, and n. To be able to extract worst case interpolator for a particular dataset $\mathcal{S}_{train}$, we use Gurobi Optimization, LLC (2021) (Gurobi Optimizer) to solve the optimization problem that follows.

$$\begin{aligned} \arg\max_{w} \quad & ||\alpha - w||_2^2 \\ \text{s.t.} \quad & w^T(x_i y_i) \geq 0, \forall (x_i, y_i) \in \mathcal{S}_{train} \\ & ||w||_2 = 1 \end{aligned} \tag{2.6.0.1}$$

Since $||\alpha - w||^2 = ||\alpha||^2 + ||w||^2 - 2\alpha^T w$, $\forall w \in \mathcal{S}^{d-1}$, and $||\alpha||_2 = ||w||_2 = 1$, we can rewrite the optimization problem as:

$$\begin{aligned} \arg\min_{w} \quad & \alpha^T w \\ \text{s.t.} \quad & w^T(x_i y_i) \geq 0, \forall (x_i, y_i) \in \mathcal{S}_{train} \\ & ||w||_2 = 1 \end{aligned} \tag{2.6.0.2}$$

Furthermore, considering α is $(1, 0, \cdots, 0)$, we can refine the optimization problem even further in order to obtain the final optimization problem that we use to find the worst case interpolator.

$$\begin{aligned} \arg\min_{w} \quad & w_1 \\ \text{s.t.} \quad & w^T(x_i y_i) \geq 0, \forall (x_i, y_i) \in \mathcal{S}_{train} \\ & ||w||_2 = 1 \end{aligned} \tag{2.6.0.3}$$

Our goal is to analyse the cumulative distribution function (with respect to probability measure $\mathbb{P}(\cdot|\mathcal{H}_{inter})$) of a random variable which represents L_2 distance of random interpolator from the signal α. In order to do so, we define a sequence of quantities which we use together with simulations in order to try and depict the behaviour of the cumulative distribution function itself. The first of those quantities is the **cumulative error function** $R_{\mathcal{S}_{train}} : [0, 2] \to [0, 1]$ which represents the percentage of interpolators on a fixed set of samples $\mathcal{S}_{train}$ that have a loss less than or equal to a particular value.

$$R_{\mathcal{S}_{train}}(\cdot) = \mathbb{P}(L_{\mathcal{D}}(w) \leq \cdot | w \in \mathcal{H}_{inter}) \tag{2.6.0.4}$$

In above definition, $L_{\mathcal{D}}$ is as defined in *Context* section. Additionally, $\mathcal{H}_{inter}$ is such that it is built based on $\mathcal{S}_{train}$. Loss function used for this problem is L_2 distance between signal α and estimator w.

Furthermore, we define **mean cumulative error function** $R_n^{mean} : [0, 2] \to [0, 1]$, as being an averaged version of the **cumulative error function**, where averaging is done over multiple instances of $\mathcal{S}_{train}$ (each instance has n number of samples in total). Averaging results in an integral over the probability measure of all possible interpolator sets $\mathcal{H}_{inter}$ (yielded by possible training sets $\mathcal{S}_{train}$) as follows:

$$R_n^{mean}(\cdot) = \int_{\mathcal{H}_{inter}} \mathbb{P}(L_{\mathcal{D}}(h_w) \leq \cdot | h_w \in \mathcal{H}_{inter}) \mathbb{P}(\mathrm{d}\mathcal{H}_{inter}) \tag{2.6.0.5}$$

For the purpose of analysing the behaviour of typical interpolators, when compared to the worst case interpolator, we will use quantiles of the mean cumulative error function for $p = 0.001$ and $p = 0.999$. To be able to do this, we define **mean quantile error function** $Q_n^{mean} : [0, 1] \to [0, 2]$ as follows:

$$Q_n^{mean}(\cdot) = \inf\{l \in [0, 2] \text{ s.t. } R_n^{mean}(l) \geq \cdot\} \tag{2.6.0.6}$$

In order to try and depict the behaviour of the mean cumulative error function for various values of ζ, d, and n, while at the same time taking into consideration the computational burden of calculating the mean quantile error function Q_n^{mean} for a particular value $q \in [0, 1]$, we will be using simulations to try and estimate values of Q_n^{mean} only for the arguments 0.001 and 0.999. Apart from that, we will approximate the value of E_{dist} (as defined in typical interpolator analysis). If we can sample iid. $w_1, w_2, \ldots, w_N \sim \mathbb{P}(\cdot|\mathcal{H}_{inter})$, then the following holds (Law of Large Numbers):

$$E_{dist} = \mathbb{E}[||w - \alpha||_2 | \mathcal{H}_{inter}] \approx \frac{\sum_{i=1}^{N} \|w_i - \alpha\|_2}{N} \tag{2.6.0.7}$$

The same also holds if we sample $w_1, w_2, \dots, w_N$ using MCMC approach, although the convergence of (2.6.0.7) is slower. Sampling of iid. interpolators, using traditional Monte Carlo approach of drawing $w \sim \mathbb{P}(\mathcal{H})$ and rejecting it if $w \notin \mathcal{H}_{inter}$, would be infeasible in high dimensions. Therefore, we sample $w_1, w_2, ..., w_N$ using MCMC LIN-ESS algorithm presented in Gessner, Kanjilal, and Hennig (2019) (code on https://github.com/alpiges/LinConGauss) to efficiently sample from $P(\cdot|\mathcal{H}_{inter})$. This method uses the MCMC approach which uses the fact that sampling from $\mathbb{P}(\cdot|\mathcal{H}_{inter})$ is equivalent to sampling from linearly constrained gaussian distribution. This is the case for distribution $\mathbb{P}(\cdot|\mathcal{H}_{inter})$, and is explained below, analogously to how they do it in Theisen et al. (2021).

$$\begin{aligned} &\mathbb{P}(\cdot|\mathcal{H}_{inter}) = \mathbb{P}(\cdot|\gamma_n = 1), \\ \text{where } \gamma_N = &\prod_{(x_i,y_i)\in\mathcal{S}_{train}} 1\{y_i sign(\langle w, x_i\rangle) \geq 0\} = \prod_{(x_i,y_i)\in\mathcal{S}} 1\{y_i\langle w, x_i\rangle \geq 0\} \\ = &\prod_{(x_i,y_i)\in\mathcal{S}} 1\{y_i(w^T x_i) \geq 0\} = \prod_{(x_i,y_i)\in\mathcal{S}} 1\{w^T(x_i y_i) \geq 0\} \end{aligned} \tag{2.6.0.8}$$

Next, our goal is to estimate the mean error quantile function $Q_n^{mean}(\cdot)$. In order to do so, we first define **empirical cumulative error function** $\hat{R}^m_{\mathcal{S}_{train}} : [0,2] \to [0,1]$ which is built using m samples $w_1, w_2, \cdots, w_m \sim \mathbb{P}(\cdot|\mathcal{H}_{inter})$ ($\mathcal{H}_{inter}$ is set of interpolators yielded by training set $\mathcal{S}_{train}$ of size n) generated by algorithm MCMC LIN-ESS (as explained above).

$$\hat{R}^m_{\mathcal{S}_{train}}(\cdot) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{L_{\mathcal{D}}(w_i) \leq \cdot\}, \tag{2.6.0.9}$$

Furthermore, we define **empirical mean cumulative error function** $\hat{R}_n^{mean,m,m_1,,\cdots,m_m}$: $[0,2] \to [0,1]$ which is the averaged version of the empirical cumulative error function. Averaging is done over m independent instances of the sample set $\mathcal{S}_{train}$ (each of size n) that induces the interpolator set $\mathcal{H}_{inter}$.

$$\begin{aligned} \hat{R}_n^{mean,m,m_1,,\cdots,m_m}(\cdot) &= \frac{1}{m}\sum_{j=1}^{m} \hat{R}^{m_j}_{\mathcal{S}_{train,j}} \\ &= \frac{1}{m}\sum_{j=1}^{m} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{L_{\mathcal{D}}(w_{i,j}) \leq \cdot\} \end{aligned} \tag{2.6.0.10}$$

Finally, we define **empirical mean error quantile function** $\hat{Q}_n^{mean,m,m_1,,\cdots,m_m} : [0,1] \to [0,2]$ as follows:

$$\hat{Q}_n^{mean,m,m_1,,\cdots,m_m}(\cdot) = \inf\{l \in [0,2] \text{ s.t. } \hat{R}_n^{mean,m,m_1,,\cdots,m_m}(l) \geq \cdot\} \tag{2.6.0.11}$$

We use the empirical mean quantile error function (Equation (2.6.0.11)) to estimate the mean quantile error function (Equation (2.6.0.6)). The estimation goes as follows. For each $j \in \{1, 2, \cdots, m\}$, we first create a train dataset $\mathcal{S}_{train,j}$ of size n which then yields an appropriate interpolator set $\mathcal{H}_{inter,j}$. After that, we use the MCMC LIN-ESS algorithm to sample m_j samples from $\mathbb{P}(\cdot|\mathcal{H}_{inter,j})$. When doing the estimation, we use $m = 100$ and $m_1 = m_2 = \cdots = m_m = 100$ in particular. Furthermore, we take into consideration only each 10^{th} sample returned by the MCMC algorithm. This way, we try and lower the dependance between samples. Next, we argue the appllicability of our estimators presented above. First, using the strong law of large numbers for Markov chains, for all training sets $\mathcal{S}$ of size n, the following holds:

$$\lim_{m\to\infty} \hat{R}^m_{\mathcal{S}}(x) = R_{\mathcal{S}}(x) \text{ a.s., } \forall x \in [0,2] \tag{2.6.0.12}$$

Using the strong law of large numbers and (2.6.0.12), it holds that:

$$\lim_{\substack{m\to\infty \\ m_j\to\infty, \forall j\in\{1,2,\cdots,m\}}} \hat{R}^{mean,m,m_1,\cdots,m_m}_n(x) = R^{mean}_n(x) \text{ a.s., } \forall x \in [0,1] \tag{2.6.0.13}$$

Trivially, from (2.6.0.13) it follows that:

$$\lim_{\substack{m\to\infty \\ m_j\to\infty, \forall j\in\{1,2,\cdots,m\}}} \hat{Q}^{mean,m,m_1,,\cdots,m_m}_n(x) = Q^{mean}_n(x) \text{ a.s., } \forall x \in [0,1] \tag{2.6.0.14}$$

We are also interested in how worst case interpolators behave. Building on optimization problem (2.6.0.3), we define the **worst case error function** on the sample set $\mathcal{S}_{train}$:

$$R_{\mathcal{S}_{train},worst} = L_{\mathcal{D}}(w), \tag{2.6.0.15}$$
$$\text{where } w \text{ is solution to the optimization problem (2.6.0.3).}$$

Furthermore, we also define the **mean worst case error function**, which is the averaged version of the worst case error function above. Averaging is done over all possible sample sets $\mathcal{S}_{train}$ (each set has n number of samples in total) and the corresponding interpolator set $\mathcal{H}_{inter}$ and probability measure.

$$R^{mean}_{worst,n} = \int_{\mathcal{H}_{inter}} R_{\mathcal{S}_{train},worst}\mathbb{P}(\mathrm{d}\mathcal{H}_{inter}) \tag{2.6.0.16}$$

Finally, we estimate the mean worst case error function using the **empirical mean worst case error function**. The applicability of the approach is based on the strong law of large numbers.

$$\hat{R}^{mean,m}_{worst,n} = \frac{1}{m}\sum_{i=1}^{m} R_{\mathcal{S}_{train,i},worst} \tag{2.6.0.17}$$

In estimator (2.6.0.17), all the sample sets $\mathcal{S}_{train,i}$,$i = 1, \cdots, m$, are drawn independently from the model defined in Definition 1.4.0.1.

After presenting all the tools needed to make the experiments and see how typical and worst case interpolators behave for different values of $\zeta = \frac{n}{d}$, n, and d, we present results of the simulations in Figures 2.6, 2.7, 2.8, 2.10. In Tables 2.1 and 2.2, we give more detailed results regarding the simulations for $\zeta = 0.1$ presented in Figure 2.7. We do that in order to make it easier to understand some of our observations regarding the simulations in general. The results of the simulations are very well aligned with the results presented in the paper Theisen et al. (2021). There, they show that as dimension d (or number of features) increases, the error rate of the typical interpolators tends to converge to the value $\epsilon(\zeta)$. This is also reflected in our experiments where with an the increasing dimension d,

	Typical interpolator		
	0.001 error percentile	0.999 error percentile	mean error value
$d = 5$	0.2350	1.9615	1.2572
$d = 10$	0.5451	1.8945	1.3460
$d = 100$	1.1351	1.5775	1.3690
$d = 250$	1.2188	1.4976	1.3689
$d = 500$	1.2774	1.4563	1.3705
$d = 1000$	1.3062	1.4268	1.3726

Table 2.1: Table shows particular values of empirical L_2 distance error distribution for typical interpolators induced by the simulations done and presented in Figure 2.7. For more details about the simulation process refer to Figure 2.7 itself.

	Worst case interpolator		
	min error value	maximum error value	mean error value
$d = 5$	1.5990	1.99993	1.9263
$d = 10$	1.8472	1.99994	1.9761
$d = 100$	1.9448	1.99491	1.9765
$d = 250$	1.9580	1.98844	1.9754
$d = 500$	1.9570	1.98511	1.9742
$d = 1000$	1.9640	1.98219	1.9741

Table 2.2: Table shows particular values of empirical L_2 distance error distribution for worst case interpolator induced by the simulations done and presented in Figure 2.7. For more details about the simulation process refer to Figure 2.7 itself.

typical interpolators tend to behave more and more similar, and their error shows signs of convergence to a particular value $\epsilon \in [0, 2]$, which depends on the ratio $\zeta = \frac{n}{d}$. As one would expect, as ζ increases, ϵ gets lower. The reason for that is that since we are working with a noiseless version of the halfspace learning model, as we add more and more data to the training set, we will cut off more and more bad interpolators (those that are at the worst case boundaries of the interpolator set $\mathcal{H}_{inter}$). It is important to note that the convergence behaviour is in accordance to what we have proved in the special case of $n = 1$ in Section 2.4. Another thing that seems to be consistent throughout all of the simulations is that as dimension d increases, mean value of the typical interpolator error gets bigger. This can be nicely seen by looking at the plots or at the results presented in the Table 2.1. By looking into the worst case interpolators error plots, we notice a similar behaviour to those of typical interpolators. Worst case interpolators also seem to converge in the sense of the error rate to value ϵ' that depends on the ratio $\zeta = \frac{n}{d}$. An important thing to mention here is that for each run of the simulation, we have only one worst case interpolator error value, while there exist uncountably many typical interpolator error values. Therefore, convergence of the worst case interpolator error rate as dimension d grows means that simulation runs tend to become more deterministic in a sense of what the worst case error value is. As $\zeta = \frac{n}{d}$ grows, the error rate of the worst case interpolators gets lower. The reasoning behind that is the same as for the similar behaviour of typical interpolators. As opposed to the case of typical interpolators, there doesn't seem to be a trend in the mean worst case error value as dimension d grows.

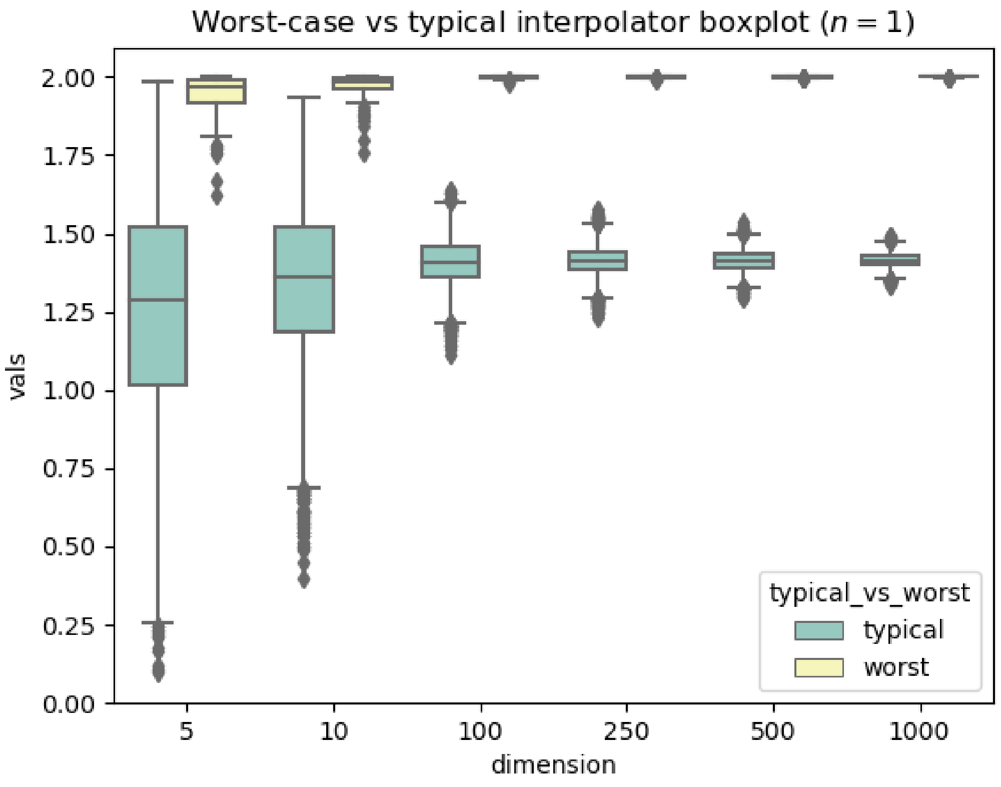


Figure 2.6: Figure shows results of the simulations from the halfspace learning model presented in 1.4.0.1. Simulations are done according to the explanaitions in the text. We repeatedly ran the simulations for dimensions of $d = 5$, 10, 100, 250, 500, and 1000, and with the number of training samples $n = 1$. Boxplot measurements of **typical interpolator's** L_2 distance from signal α are plotted with a green color. Boxplot of the worst case interpolator's (obtained using the optimization procedure 2.6.0.3) L_2 distance from signal α is plotted with a yellow color.

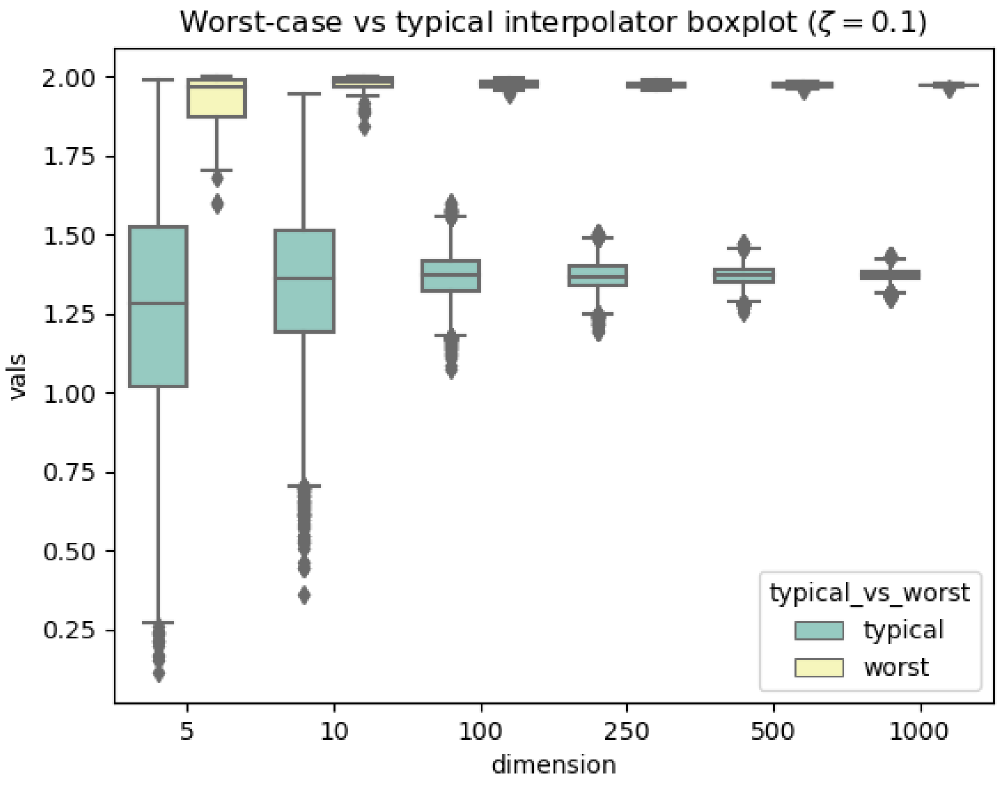


Figure 2.7: Figure shows results of the simulations from the halfspace learning model presented in 1.4.0.1. Simulations are done according to the explanaitions in the text. We repeatedly ran the simulations for dimensions of $d = 5$, 10, 100, 250, 500, and 1000, and with $\zeta = \frac{n}{d} = 0.1$. Boxplot measurements of **typical interpolator's** L_2 distance from signal α are plotted with a green color. Boxplot of the worst case interpolator's (obtained using the optimization procedure 2.6.0.3) L_2 distance from signal α is plotted with a yellow color.

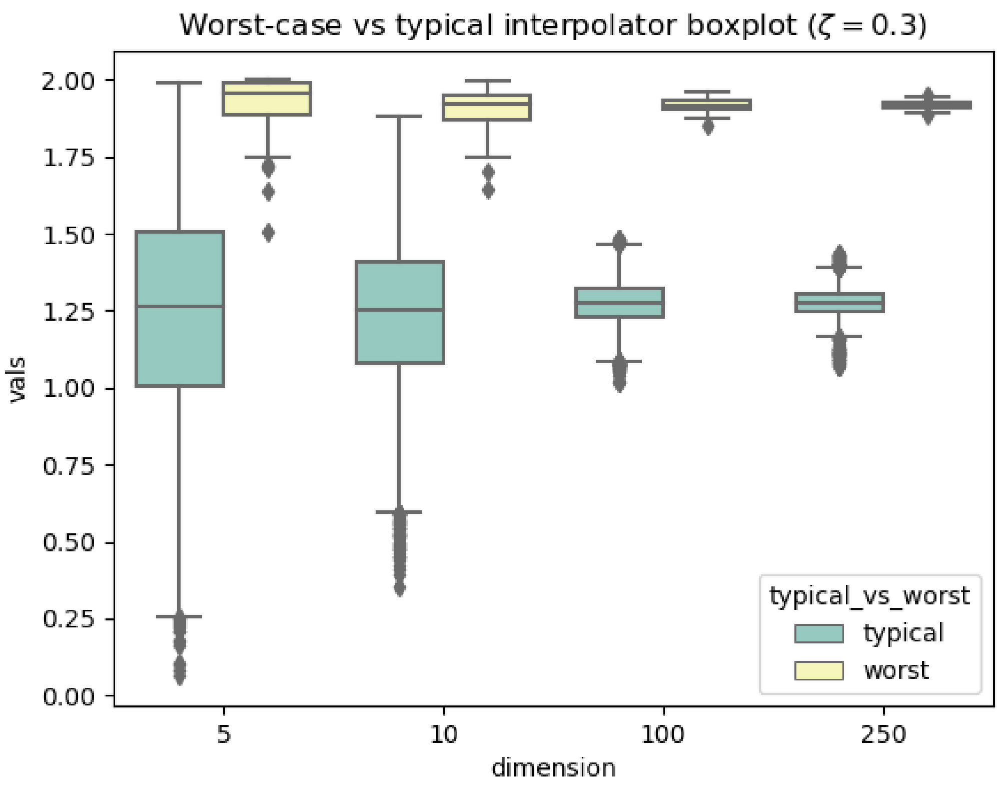


Figure 2.8: Figure shows results of the simulations from the halfspace learning model presented in 1.4.0.1. Simulations are done according to the explanaitions in the text. We repeatedly ran the simulations for dimensions of $d = 5, 10, 100, 250$, and with $\zeta = \frac{n}{d} = 0.3$. Boxplot measurements of **typical interpolator's** L_2 distance from signal α are plotted with a green color. Boxplot of the worst case interpolator's (obtained using the optimization procedure 2.6.0.3) L_2 distance from signal α is plotted with a yellow color.

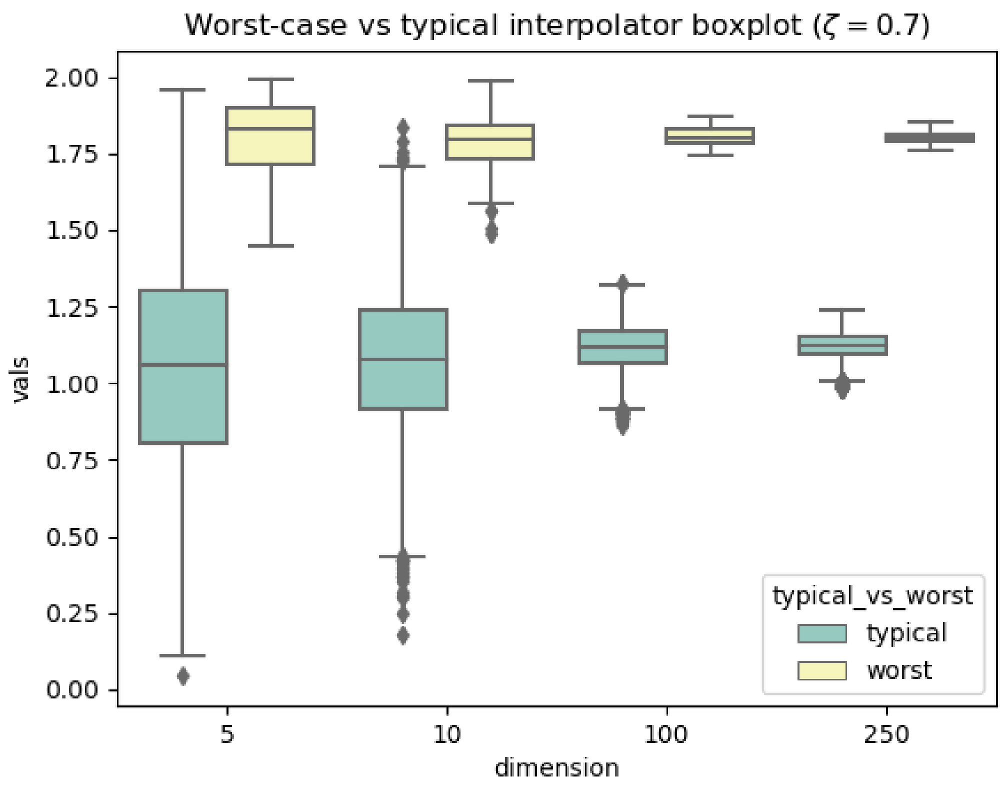


Figure 2.9: Figure shows results of the simulations from the halfspace learning model presented in 1.4.0.1. Simulations are done according to the explanaitions in the text. We repeatedly ran the simulations for dimensions of $d = 5$, 10, 100, 250, and with $\zeta = \frac{n}{d} = 0.7$. Boxplot measurements of **typical interpolator's** L_2 distance from signal α are plotted with a green color. Boxplot of the worst case interpolator's (obtained using the optimization procedure 2.6.0.3) L_2 distance from signal α is plotted with a yellow color.

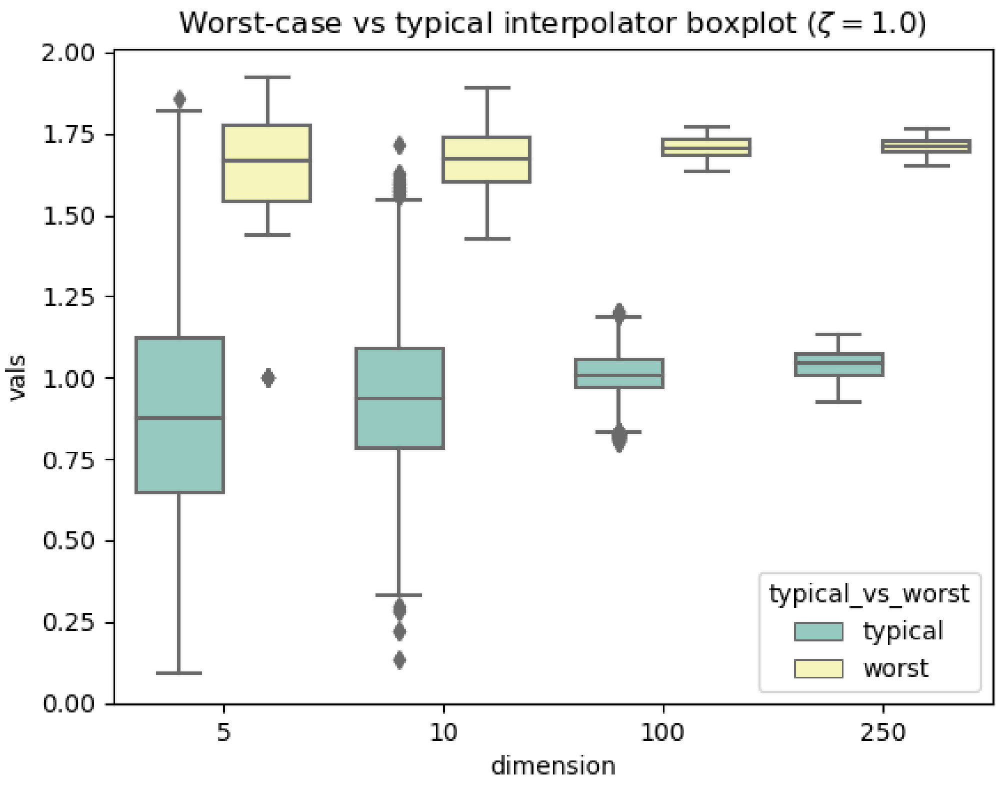


Figure 2.10: Figure shows results of the simulations from the halfspace learning model presented in 1.4.0.1. Simulations are done according to the explanaitions in the text. We repeatedly ran the simulations for dimensions of $d = 5$, 10, 100, 250, and with $\zeta = \frac{n}{d} = 1.0$. Boxplot measurements of **typical interpolator's** L_2 distance from signal α are plotted with a green color. Boxplot of the worst case interpolator's (obtained using the optimization procedure 2.6.0.3) L_2 distance from signal α is plotted with a yellow color.

Chapter 3

Neural networks

In this chapter we compare results obtained from the halfspace learning model analysis in Chapter 2 to the results that would be given for the same model by the existing neural network guarantees based on the uniform convergence framework. Additionally, before doing the comparison, we also argue why it makes sense for us to actually do so.

3.1 Comparison to neural networks generalisation guarantees

3.1.1 Preliminaries

Noiseless version of the halfspace learning model presented in Definition 1.4.0.1 (that we also analysed in Chapter 2) can be represented as a 1-layer neural network with the *sign* activation function in the output layer. $x \sim \mathbb{N}(0, I)$ serves as an input, while $y = sign(\langle x, \alpha \rangle)$ represents target label. α stands for the weights of the neural network that we try to estimate, given the training dataset $\mathcal{S}_{train}$. In neural network terminology, we try to learn the weights w that are as close as possible to the real signal α. Next, we make a few changes to the presented neural network in order to make it able to learn using the optimisation algorithms based on the gradient of the underlying loss function. While describing the changes, we will also argue why we are allowed to make them, and how do these changes keep the equivalence between our starting halfspace learning model and the resulting neural network. The first change we make is replacing the $f(t) = sign(t)$ activation function with the *logistic* activation function $logist_\lambda(t) = \frac{1}{1+e^{-\lambda t}}$. We do the change in order to have the activation function, which is differentiable w.r.t. to its argument. This will later allow us to optimise the (differentiable) objective function which will be presented in what follows. Furthermore, as $\lambda \to \infty$, $logist_\lambda(t)$ converges to $sign(t)$, $\forall t \in \mathbb{R}$. With the changes introduced, function $g : \mathcal{R}^d \to \mathcal{R}$ that our neural network calculates for input data x is $g(x) = logist_\lambda(x^T w)$, where w are the weights of the neural network. Rewriting the expression a bit, we get the following: $g(x) = logist_\lambda(x^T w) = \frac{1}{1+e^{-\lambda(x^T w)}} = \frac{1}{1+e^{-x^T(\lambda w)}}$. Replacing λw with $w^{'}$, this can be further rewritten as $g(x) = \frac{1}{1+e^{-x^T w^{'}}} = logist_{\lambda=1}(x^T w^{'})$, where $w^{'}$ are the weights of the neural network. What this means is that we can use the logistic function with $\lambda = 1$. By scaling the weights w with constant $\lambda > 1$ we get the same final model that as norm of the weights w grows, behaves more and more like the model with the *sign* activation function. Therefore, without the loss of generality, we can assume $\lambda = 1$ and we will not change anything with respect to the model behaviour. For large L_2 norm of weights w, neural network will faithfully replicate the behaviour of the original neural network with the *sign* activation function. The second change we make is related to the fact that our goal is to use the neural network together with existing generalisation gaurantees for neural networks. In particular, we will be using it with the generalisation gurarantees presented in Neyshabur, Bhojanapalli, McAllester, and Srebro (2017) and Bartlett, Foster, and Telgarsky (2017). The guarantees we will use requires the user to be able to observe the margin of the neural network. Having the logistic activation function does not allow one to have the direct control over the margin. Therefore, instead of it, we will be using the 2-dimensional output neural network with the *softmax* activation function. *Softmax* function for $t \in \mathbb{R}^k$ is defined as a mapping $\mathbb{R}^k \to \mathbb{R}^k$, where $f(t)_i = \frac{\exp t_i}{\sum_{j=1}^{k} \exp t_j}$. We are allowed to do the exchange between the 1-dimensional *logistic* output with the 2-dimensional *softmax* output because the expresiveness of the two versions of the neural network architectures stays the same. Finally, we use the *cross-entropy* loss function as the objective function that we minimize (over all $(x_i, y_i) \in \mathcal{S}_{train}$) when training our final version of the neural network. *Cross-entropy* loss function is defined as $CE(\mathcal{S}_{train}) = \sum_{i=1}^{n}(-log(f_{y_i}(x_i)))$, where $f_{y_i}(x_i)$ is the output of the neural network corresponding to the probability that the class of current input x_i is $y_i \in \{0, 1\}$. A convenient thing with the *cross-entropy* loss function is that it allows us to control the margin of the neural network. The only thing left to explain is what the margin of the neural network actually is. The margin of the neural network (with *softmax* activation function) at the datapoint (x, y) is defined as $\Gamma(x, y) = f(x)_y - \max_{y' \neq y} f(x)_{y'}$, where $f(x)$ is the *logit* function (quantity that gets fed into the *softmax* layer in order to get the probability distribution over the possible classes). In our case of the 2-dimensional output (where $y \in \{0, 1\}$), margin of the neural network is defined as $\Gamma(x, y) = f(x)_y - f(x)_{1-y}$. Graphical representation of the final neural network model that we will be using, as implemented in the *torch* library and generated with Adam Paszke, Soumith Chintala, Anton Osokin, Will Price, soulitzer, albanD (2021) (PyTorchViz library), which is depicted in Figure 3.1.

3.1.2 Generalisation guarantees comparison

In this section, using the equivalence between our final neural network model and the halfspace learning model (as explained in the previous section), we take the results rigurously obtained for the halfspace learning model in Chapter 2 and compare it to the generalisation bounds obtained for the neural networks under the same setting. In particular, this means that we will be looking into the behaviour of the guarantees for the special case when the number of training samples $n = 1$. Prior to training the neural networks, we normalise the input values $x_i, i = 1, \cdots, n$, so that $\forall(x_i, y_i) \in \mathcal{S}_{train}$, $||x_i||_2 = B = 1$. The reason we normalise the inputs is that the only information that is important about the inputs (x values) (when coupled with y values) is its direction, not its norm. Therefore, since normalisation does not change the information gained from (x, y) pairs, we can normalize it without concern. Another reason why we do so is that it allows us to keep the importance between the samples the same when training the neural network. The final reason we do that is that the way the generalisation guarantee in Neyshabur et al. (2017) is laid out is that it involves the upper bound on the norm of all the input data from the training set. Therefore, by doing the normalisation, we inherently set the value of the upper bound to the deterministic value of 1 that we can directly use in the bound. When we do the training, we train the neural networks until $\Gamma(x, y) \geq 10, \forall(x, y) \in \mathcal{S}_{train}$. This is the place where we actually utilise the fact that we use the *softmax* activation function instead of the *logistic* activation function. In the two guarantees that we use,

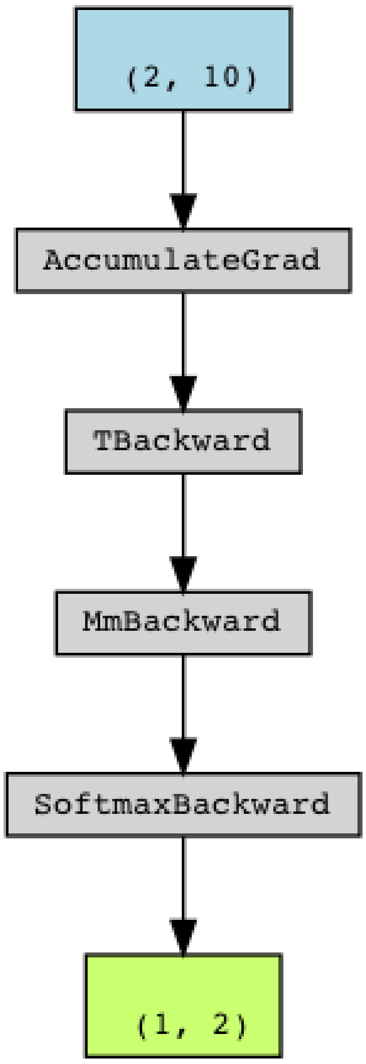


Figure 3.1: PyTorchViz visualization of the neural network (with $softmax$ output) version of the halfspace learning model for the input dimension $d = 10$. The blue square represents the weights in the first layer. As it is apparent from the graph, there is no separate square for the bias, and dimension of the weight is 10×2, resulting with the 2-dimensional first layer. After that comes the *Softmax* layer which can be seen on the lowest grey square. This results in ending up with the output of dimension 2, as denoted on the green square.

first terms in both are laid out in terms of the margins over the training dataset that we control by training the neural network until it is at least 10. The final important thing to note is that generalisation bounds laid out in terms of margin on $(x, y) \sim \mathcal{D}$ have to be nonpositive. This is just a different way of saying what the probability to misclassify the data point (x, y) sampled from data distribution $\mathcal{D}$ (which in our case is the halfspace learning model from Definition 1.4.0.1) is. Next, we make the actual comparisons to the existing guarantees. First, we take a look at the bound in Theorem 1 of Neyshabur et al. (2017). There, they give the bound on the probability of misclassification of the neural network as follows:

$$P(\Gamma(x,y) \leq 0) \leq \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_{train}} \mathbb{1}\{\Gamma(x,y) \leq \gamma\} \\ +4\sqrt{\frac{\frac{42^2 d^2 B^2 \tilde{\beta}^{2d-2} h \log 4hd}{2\gamma^2} \sum_{i=1}^{d} ||W_{i,learned} - W_{i,init}||_F^2 + \log \frac{6n}{\delta}}{n-1}} \tag{3.1.2.1}$$

The first term of the bound can be made arbitrarily low since we can manually set the value of γ. In particular, we set γ equal to 10, and by doing so, we inherently make the first term of the bound (3.1.2.1) equal to 0. In the denominator of the second term there is a value $n-1$ which equals 0 for the case when n is 1, and that makes the bound completely inappropriate to be applied to the case when the number of training samples n is 1.

Another bound we compare our results to is the bound from Lemma A.9. in Bartlett et al. (2017). The bound is stated as follows:

$$P(\Gamma(x,y) \leq 0) \leq \frac{1}{n} \sum_{(x,y) \in \mathcal{S}_{train}} \mathbb{1}\{\Gamma(x,y) \leq \gamma\} + \frac{8}{n} \\ + \frac{144 log(n) \log(2W)}{\gamma n} (\prod_i \phi_i)(1 + ||X||_2) A \\ + \sqrt{\frac{9}{2n}} \sqrt{log(\frac{1}{\delta}) + log(\frac{2n}{\gamma}) + B} \tag{3.1.2.2}$$

$$\text{where } A = (\sum_{i=1}^{L} ((\frac{1}{L} + ||A_i^T - M_i^T||_{2,1}) \prod_{j \neq i} (\frac{1}{L} + ||A_j||_\sigma))^{2/3})^{3/2}, \tag{3.1.2.3}$$

$$B = 2log(2 + ||X||_2) + 2\sum_{i=1}^{L} (log(2 + L||A_i^T - M_i^T||_{2,1}) + log(2 + L||A_i||_\sigma)) \tag{3.1.2.4}$$

Equivalently to what was stated for the first term in the bound (3.1.2.1), we set it to 0 by simply setting the value of γ equal to 10. By simple insertion of $n = 1$ into the second term, we find that it equals 8. The third term equals 0 because of the $log(n) = log(1) = 0$ term. The fourth term is also greater than or equal to zero. This is because $\sqrt{\frac{9}{2n}} = 4.5 > 0$, $log(\frac{1}{\delta}) \geq 0$, since δ is an arbitrary value chosen from $(0, 1]$. This implies that $log(\frac{2n}{\gamma}) = log(\frac{1}{5}) = -log(5)$, but this negative term is already cancelled by the first summand of the right hand side of (3.1.2.4) which equals $2log(2 + ||X||_2) = 2log(2 + 1) = 2log(3) = log(9) \geq log(5)$. All other parts of B are greater than or equal to 0, and therefore, we end up with the fourth term in the summand being greater than or equal to 0. From the analysis of the summands, we come to the conclusion that the guarantee provides the following upper bound on the misclassification probability: $P(\Gamma(x, y) \leq 0) \leq \tau$, where τ is some value greater than or equal to 8. Therefore, the result given by this bound is not of much use since it only gives us the bound of a value greater than 1. Therefore, from what we have observed by looking at the results of the neural network guarantees, neither of the two generalisation bounds is suited to be used with our model for number of samples $n = 1$. Although this is a very special case that was suited to look into for the sake of comparison to the results we obtained in Chapter 2, it shows that there exist intrinsic problems with bounds. In particular, the one from Neyshabur et al. (2017) has the problem of not being applicable on the dataset with number of samples $n = 1$, while the other one shows vacuous behaviour when applied to our problem with n being 1. Moreover, the problems for the bounds for $n = 1$ might indicate that similar problems might ocure for more general cases. Additionally, from the experience of using the bounds for this paper, they were extremely hard and impractical to use. On the example of the halfspace learning model and $n = 1$, we showed much better applicability of the typical interpolator analysis rather than looking at the worst case bound that are given by the generalisation guarantees. Because of all that, and the problems that generalisation bounds suffer, as showed in Nagarajan and Kolter (2019), we can say that generalisation bounds might indeed be completely unsuitable for the neural networks and that typical interpolator analysis might be a more suitable alternative. As we did not explain in detail the meaning of all the terms used in generalisation guarantees, we do so in the Appendix. Apart from that, results obtained from the simulations of training the halfspace learning dataset are also present. In particular, we plot the Frobenius norm of the difference between the starting and learned neural network weights.

Chapter 4

Conclusion

4.1 Motivation

In this paper we mainly built on the paper Theisen et al. (2021) and their work on the statistical mechanics approach to solving the problem of uniform convergence framework failure when applied on a complex hypothesis space such as neural networks. In their work, they provably showed results for the behaviour of a typical interpolator in the case of the halfspace learning model (1-layer neural networks). They do so under the assumption that training and testing samples have a fixed positive correlation $\sigma \in (0, 1]$ with each other. These assumption seemed to be unrealistically artificial so we investigated the behaviour of a typical and the worst case interpolator under a much more realistic scenario of having a general half-space learning model (as presented in Definition 1.4.0.1).

4.2 Typical vs worst case interpolator

For such a model (noiseless version), and the case when number of training samples n is 1, we provably showed convergence of behaviour of both typical and worst case interpolators. As dimension grows to infinity ($d \to \infty$) typical interpolators showed behaviour of convergence to a random classifier (in a sense that classification error rate converges to 0.5), while worst case interpolator classification error converges to value of 1. In addition to that, we conducted simulations for a more general case when number of training samples n is greater than 1. Results were consistent with theoretical findings for n being 1. In particular, as dimension grows, and $\zeta = \frac{n}{d}$ is kept constant, typical interpolators tend to converge (in a sense of error rate) to a specific value $\epsilon(\zeta) \leq 0.5$. At the same time, worst case interpolator error rate converges to a particular value $\epsilon'(\zeta) > \epsilon(\zeta)$. It would be very interesting, although probably extremely hard, to provably show the behaviour of typical and worst case interpolators for the case when number of samples n is greater than 1. Additionally, we studied only the noiseless version of the halfspace learning model. Therefore, it would be interesting to extend our results by looking at the case of halfspace learning model with the noise.

4.3 Comparison to the neural network guarantees

Using the (approximate) equivalence between the halfspace learning model and 1-layer neural network classification architecture, we further compared the existing neural network guarantees to the findings (obtained both provably and using the simulations) on performance of halfspace learning model interpolators (as explained in the section above). By doing the comparison, we found that convergence guarantees not only give much worse guarantees on performance of neural networks when compared to our results, but they also give useless guarantees such as telling the user that classification rate will be smaller than some value $k > 1$. This all provides support for the statement that uniform convergence framework fails when applied naively on the neural networks. Neural networks that we analyzed in this work include simple 1-layer neural network models. Therefore, it seems like a natural continuation of work to try and generalise results on a topic of typical vs worst case interpolators to much more general neural networks with more than 1 layer and with various activation functions. Additionally, there seems to be a close connection between the random features neural network models and typical interpolators studied here. This exact connection could be studied, and might give further insights on why deep neural networks work as well as they do.

Bibliography

Adam Paszke, Soumith Chintala, Anton Osokin, Will Price, soulitzer, albanD (2021). PyTorchViz.

Bartlett, P., D. J. Foster, and M. Telgarsky (2017). Spectrally-normalized margin bounds for neural networks.

Datta, L. (2020). A survey on activation functions and their relation with xavier and he normal initialization. *CoRR abs/2004.06632*.

Gessner, A., O. Kanjilal, and P. Hennig (2019). Integrals over gaussians under linear domain constraints. *CoRR abs/1910.09328*.

Gurobi Optimization, LLC (2021). Gurobi Optimizer Reference Manual.

Hopcroft, J. and R. Kannan. Computer science theory for the information age. https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/hopcroft-kannan-feb2012.pdf.

Nagarajan, V. and J. Z. Kolter (2019). Uniform convergence may be unable to explain generalization in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems*, Volume 32. Curran Associates, Inc.

Neyshabur, B., S. Bhojanapalli, D. McAllester, and N. Srebro (2017). A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *CoRR abs/1707.09564*.

Plan, Y. and R. Vershynin (2013). Dimension reduction by random hyperplane tessellations.

Theisen, R., J. M. Klusowski, and M. W. Mahoney (2021). Good classifiers are abundant in the interpolating regime.

Vershynin, R. (2018). *Random Vectors in High Dimensions*, pp. 3869. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

Chapter 5

Appendix

5.1 Code

Code used during the project can be found at the following link:
https://github.com/lk49108/master_thesis

5.2 Neural networks

In this section we add additional information regarding the meaning of the terms in neural network guarantees (3.1.2.1), and (3.1.2.2). Furthermore, as the guarantee in (3.1.2.1) contains the term with the Frobenius norm of the difference between the starting and the learned weights, we plot the simulation results for $n = 1$.

5.2.1 Terms in the guarantee (3.1.2.1)

d corresponds to the number of layers which for our model is 1. h stands for the upper bound on a number of neurons in each layer of the neural network (not including the input layer). In our case h is 2, and since $d = 1$, $\tilde{\beta}^{2d-2} = 1$.

5.2.2 Terms in the guarantee (3.1.2.2)

L stands for the number of layers, which is 1 in our case. ϕ_i stands for the Lipschitz constant of a nonlinearity used in the layer i. Since we only have one layer that is linear, in our case $\phi_1 = 1$. Norm of the dataset $||X||_2$ equals to $\sqrt{\sum_{i=1}^{n} ||x_i||_2^2}$. Since we work under the setting when n is 1, and we normalise x_1 such that $||x_1||_2 = 1$, we have: $||X||_2^2 = 1$. A_i stands for the weights in the layer i. M_i is a parameter which can be set to an arbitrary value and stands for the reference matrix in the layer i. We set it to 0 matrix (matrix whose elements all equal 0). δ is a parameter that can be set to an arbitrary value in the range $(0, 1]$, and tells us the level of probability that the guarantee fails at. Therefore, it is usually set to some small value of around 0.01 (corresponding to 1% chance for the bound to give us a bad guarantee). $|| \cdot ||_{2,1}$ stands for the $(2, 1)$ matrix norm, while $|| \cdot ||_\sigma$ stands for the spectral norm.

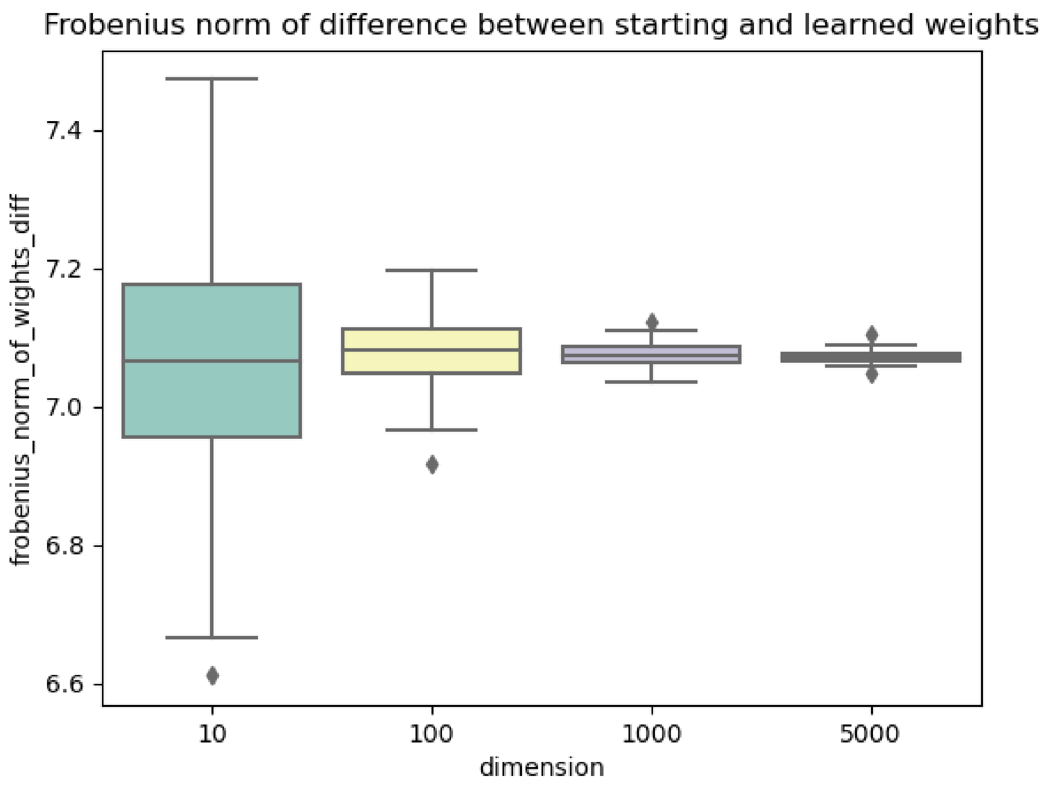


Figure 5.1: A boxplot of the Frobenius norms of the difference between learned and initial weights for different values of data dimension d. Measurements were obtained using the simulations as explained in the text. Table 5.1 gives additional information on the minimum, maximum, and mean values of the measurements.

5.2.3 Simulation results regarding the norm of the difference between the learned and the initial weights as given in the term in (3.1.2.1)

In this section we give results regarding the simulations done in order to analyse the behaviour of the Frobenius norm term in the guarantee (3.1.2.1). These simulations are under the same setting we were working with in Chapter 3. In particular, we work with the setting for $n = 1$, and $\gamma = 10$. Since in our model the number of layers d is 1, the term is as follows: $||W_{1,learned} - W_{1,init}||_F^2$. $W_{1,learned}$ corresponds to the learned weight (learning is stopped when the margin for the sample (x_1, y_1) is greater or equal to γ, which is 10), and $W_{1,init}$ corresponds to the initial weight as initialised by the torch library. Initialisation is done using the method similar to Xavier initialisation Datta (2020) where we initialise the weights using uniform distribution in the range $[-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}]$, where d is the size of the input layer. The simulation goes as follows: for each dimension $d \in \{10, 100, 1000, 5000\}$, we first sample $\mathcal{S}_{train} = \{(x_1, y_1)\}$ from the noiseless version of the halfspace learning model defined in Definition 1.4.0.1. After that, we train the network on the sample (x_1, y_1) until the margin for it exceeds the value of 10. After training is done, we compare the final weights to the initial weights by taking the difference and computing the Frobenius norm of it. Figure 5.1 shows the boxplot of the results of the simulation. Table 5.1 gives additional information on the results presented in Figure 5.1.

	Frobenius norm of change in the weights		
	Minimum value	Mean value	Maximum value
$d = 10$	6.611249	7.0702724	7.472321
$d = 100$	6.916219	7.0789757	7.1976295
$d = 1000$	7.0350823	7.074441	7.1218486
$d = 5000$	7.0479436	7.0709925	7.1039443

Table 5.1: Minimum, maximum and mean values of the measurements presented in the Figure 5.1.

Declaration of Originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor .

Title of work (in block letters):

HALFSPACE LEARNING TYPICAL INTERPOLATION AND 1-LAYER NEURAL NETWORK GUARANTEES

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
KOKOT	LEONARDO
VAN DE GEER	SARA
KUCHELMEISTER	FELIX

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the Citation etiquette information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work .
- I am aware that the work may be screened electronically for plagiarism.
- I have understood and followed the guidelines in the document *Scientific Works in Mathematics.*

Place, date:	**Signature(s):**
Zürich, September 10th 2021	Leonardo Kokot

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.