Computational Learning Theory aspects of Piecewise Polynomial and Sigmoidal Neural Networks

Dissertation Submitted in Partial Fulfilment of the Requirements for the Degree of

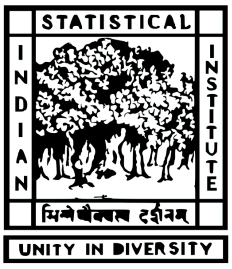
Master of Technology
in
Computer Science

by

Soumya Kanti Das
[Roll No: CS-1729]

Under the Guidance of

Dr. Swagatam Das
Associate Professor
Electronics and Communication Sciences Unit (ECSU)



Indian Statistical Institute
Kolkata-700108, India

July 2019

To my family and supervisor

CERTIFICATE

This is to certify that the dissertation entitled **"Computation Learning Theory aspects of Piecewise Polynomial and Sigmoidal Neural Networks"** submitted by **Soumya Kanti Das** to Indian Statistical Institute, Kolkata, in partial fulfilment for the award of the degree of **Master of Technology in Computer Science** is a *bona fide* record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Swagatam Das
Associate Professor,
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108, India.

Acknowledgements

I would like to take this opportunity to thank people who are behind my success in this project.

Prima facie, I would like to thank my parents, family members and teachers who supported me in every walk of my life.

I would like to show my highest gratitude to my adviser, *Dr. Swagatam Das* of Electronics and Communication Sciences Unit, for his guidance and continuous support and encouragement. His zeal and method of teaching are highly motivating.

I would also like to thank *Dr. Mandar Mitra, Dr. Utpal Garain and Dr Debapriyo Majumdar*, for their valuable suggestions and discussions.

My deepest thanks to all the professor of Indian Statistical Institute, for their valuable suggestions which added an important dimension to my research work.

Last but not the least, I would like to thank all of my friends for their help. I would also like to thank all those, whom I have missed out from the above list.

Soumya Kanti Das
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

VC (Vapnik Chervonenkis) Dimension is a useful tool for measuring the power of a neural network or some other types of classifiers. In the field of learning theory VC dimension represents the generalized power of a neural network. From mid 20^{th} century researchers have been interested in this work and have provided a vast horizon of upper and lower bounds for VC dimension of a neural network. Most of the published work assumes feed forward neural network with no skip connections to establish the upper and lower bounds of VC dimension. In this work we establish that the upper bound of VC Dimension for neural network with piece wise polynomial activation functions can be tighter. Along with this we proposed some other methods for calculating VC Dimension upper bound for RVFLN (neural network with skip connections). Most of the relevant work on VC Dimension upper bound for neural network with sigmoidal activation functions are based on model theoretic approach or number of operations on a basic computing model. Later in this work we give a different approach for calculation of VC Dimension upper bound for neural network with sigmoidal activation functions. Moreover on top this we give an idea about how a theoretical test error rate and practical test error rate depend upon on the number of layers and the number of parameters for a feed forward neural network.

Keywords: Growth function, Shattering, VC Dimension, function approximation, bit extraction technique.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Neural Networks are the star performers of modern machine learning literature. So in theoretical machine learning, researchers mainly focus on the expressive power of a neural network. The complexity of a neural classifier depends on the number of points that can be classified correctly. VC dimension is used as a tool to compute the sample bound for statistical PAC learning and also to measure the complexity of a classifier. VC Dimensions where originally defined by Vladimir Vapnik and Alexey Chervonenkis in 1971. In PAC frame model, we need to find the minimum number of samples needed during training time such that the classifier predicts all the labels of the samples correctly. For this calculation we need VC dimension as a prerequisite tool. For model evaluation Structural Risk Minimization (SRM) technique is important in which we use VC dimension. Also in computational geometry VC Dimension is used to determine the critical parameters in the size of $\epsilon-nets$, which determine the complexity of approximation algorithms. Beside that, VC dimension also predicts a probabilistic upper bound on the test error rate of a classifier model.

In general we can say VC Dimension upper bound is more than the practical test error bound. During the inception phase of neural network, people mainly used linear or piece-wise polynomial activation function. But recently, the data and its distribution are more complex than previous, so in a classification task based on neural network we use sigmoid, $tanh$ activation functions instead of just linear function. For this reason the calculation of VC Dimension also becomes more harder than previous bounds. So instead of the usual approach, researchers proposed some model theoretic approach for calculations. As of now, we know general feed forward neural network (FNN) takes too much time during back propagation. To overcome this, some scientists discovered a type of neural network which is a combination of SLFN (Single Layer Feed Forward Neural Network) and functional link, which is famously known as RVFLN (Random Vector Functional Link Network). In this network from input layer to hidden layer the weights are randomly assigned so that during back propagation weights are not updated. And also from input layer to output layer there are skip connections. For this structural advantage RVFLN takes less time in back propagation with respect to a FNN. Researchers have proposed many bounds, among those some are tighter than others. So we did a study

which will give assurance about the trade off between these bounds. And at last we proposed a procedure to handle VC Dimension for sigmoidal neural network using algebraic topology concepts. But nowadays we see that deep learning has a great importance, so we can extend our learning theory concepts in this field. Because theoretical foundations are the basis of every practical concept.

1.2 Our Contributions

Our contributions are summarized as follows.

- In the paper [2], the authors have proposed an idea about whether $O(w \log w)$ and $O(w^2)$ bound for neural network can be made closed enough, where w is the total number of parameters of the corresponding neural network. But in the paper [18], the author constructed a neural network which upper bounded by $w \log w$. The author of the paper [2] constructed a network which achieves the bound w^2, but the network takes inputs from R^2 and R. Here we propose a construction of a neural network which takes input from R^{m}, $m \geq 2$ and shatters the same size set. This network only consists of threshold and linear gates.

- This thesis also provides a relationship between number of parameters and VC Dimension for input domain R^{m}, $m \geq 2$ of a neural network. Here this trade off depends upon the input dimension m; the constructed network contains threshold, linear, multiplication and division gates. Basically a network with multiplication and division gate comes when we are working with network with continuous (such as sigmoid) activation function.

- We made the VC upper bound tighter than the existing one for neural network with piece wise polynomial activation functions.

- We have also proposed some techniques for calculating VC upper bound for neural network with S type activation functions (mainly sigmoid and tanh).

- Also for RVFLN, we suggest an idea for calculating VC dimension upper bound.

- We have made a comparison among different upper bound on VC dimension for similar type of network (having same set of activation functions), and also for neural network with different configurations of activation functions.

- We have drawn a conclusion on the fact of relationship between theoretical test error rate and practical test error rate for a neural network with classification task. We have reached to this conclusion by performing experiments on some benchmark data sets.

1.3 Thesis Outline

Chapter 2 covers a brief details of growth function, shattering, VC dimension. Here we discussed about set theoretic as well as functional way definition of the above mentioned terms. This chapter also includes some general properties of those same. Chapter 3 contains different ideas of calculating VC upper and lower bound for different types of feed forward neural network. This chapter also elaborates importance of VC dimension on machine learning. Chapter 4 covers description of Random Vector Functional Link Network

(RVFLN). In Chapter 5, we discuss our proposed ideas and constructions. In Chapter 6 we also suggest a scope of future work on this field.

Chapter 2

Preliminaries and Background

2.1 Growth function and Shattering

We denote the sample space by $X \subset \mathbf{R^m}$ which is a collection of data points or samples. The sample space also known as input space. And collection of activation functions of a particular type known as hypothesis space which is denoted by H. The cardinality of H is finite or infinite. If it is finite we can use decision tree as a complexity measure and for infinite case we use VC (Vapnik Chervonenkis) Dimension. The VC dimension is geared towards binary classification.

Definition 2.1.1 (Growth Function)**.** *The growth function $\Pi_H : \mathbf{N} \to \mathbf{N}$ for a hypothesis set H is defined by*

$$\Pi_H(m) = \max_{(x_1,x_2,.....,x_m)\subset X} |\{(h(x_1), h(x_2), ..., h(x_m)) : h \in H\}|, \forall m \in \mathbf{N}.$$

Basically $\Pi_H(m)$ is the maximum number of ways m points can be classified using H.

Set Theoretic Definition: *Let H be a set family and C be a set, then $\Pi_H(m) := \max_{C:|C|=m} |H \cap C|$.*

Definition 2.1.2 (Shattering)**.** *2^m is the maximum number of classification of m points by H. We say that a sample space X of length m shattered by hypothesis space H if this maximum value is attained, that is H gives all possible classifications of X.*

Set Theoretic Definition: *Let S be a set family (set of sets) and C a set. Then $S \cap C := \{s \cap C | s \in S\}$. we say a set C is shattered by S if $|S \cap C| = 2^{|C|}$.*

Example 2.1.1. *No 4 element set $S \subset \mathbf{R^2}$ can be shattered by C = all open half spaces. But every non collinear three element set can be shattered.*

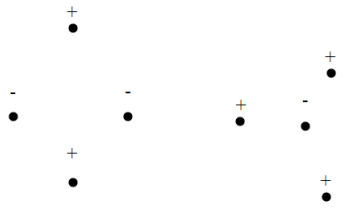


Figure 2.1: Four points can not be shattered by half spaces

2.1.1 Properties

Consider each function in function class F takes value in some finite set Y. Let $F \subset Y^X$ be a class of Y valued functions. $F_{|x_1{}^m}$ is the function class F restricted to $x_1, ..., x_m$, i,e $F_{|x_1{}^m} := \{(f(x_1), ..., f(x_m)) : f \in F\}$. $F_{|x_1{}^m}$ is finite and $|F_{|x_1{}^m}| \leq \min\{2^m, |F|\}$. Define $\Pi_H(m) := \max_{x_1{}^m \in X} |F|_{x^m{}_1}|$. Note that $\Pi_H(m) \leq |Y|^m$.

Lemma 2.1.1 ([36]). *Let $F^1 \subset Y_1^X$ and $F^2 \subset Y_2^X$ be two function classes. Let $F = F^1 \times F^2$ be their Cartesian product. Then*

$$\Pi_F(m) \leqslant {\Pi_F}^1(m).{\Pi_F}^2(m).$$

Proof. Fix x^m. By definition of cartesian product we can write

$$|F_{x^m}| = |F^1_{u^m}|.|F^2_{v^m}|$$

Now taking max in both sides, which imply

$$\Pi_F(m) \leqslant {\Pi_F}^1(m).{\Pi_F}^2(m)$$

Since x_m is arbitary, this completes the proof. □

Lemma 2.1.2 ([36]). *Let $F^1 \subset Y_1^X$ and $F^2 \subset Y_2^{Y_1}$ be two function classes. Let $F = \left(F^2 \circ F^1\right)$ be their composition. Then*

$$\Pi_F(m) \leqslant {\Pi_F}^1(m).{\Pi_F}^2(m).$$

Proof. Fix $x^m \in X^m$. By definition of F, we have

$$\begin{aligned} F_{x^m} &= \{(f_2(f_1(x_1)), f_2(f_1(x_2)), ..., f_2(f_1(x_m))) : f_1 \in F^1, f_2 \in F^2\} \\ &= \bigcup_{v \in {F^1}_{x^m}} \{(f_2(v_1), f_2(v_2), ..., f_2(v_m)) : f_2 \in F^2\} \end{aligned}$$

Now we have,

$$\begin{aligned} |F_{x^m}| &\leqslant \sum_{v \in {F^1}_{x^m}} |\{(f_2(v_1)), f_2(v_2)), ..., f_2(v_m)) : f_2 \in F^2\}| \\ &\leqslant \sum_{v \in {F^1}_{x^m}} {\Pi_F}^2(m) \\ &= |{F^1}_{x^m}|.{\Pi_F}^2(m) \\ \Pi_F(m) &\leqslant {\Pi_F}^2(m).{\Pi_F}^1(m) \end{aligned}$$

Since x^m is arbitary, this completes the proof. □

2.2 VC Dimension

Cross Validation, Bayesian Information Criteria, Structural Risk Minimization are the methods for evaluating a machine learning model. Among all these methods which model selection method is best? Understanding which learning machines are more less power full under which circumstances. To reach a conclusion we can focus on VC Dimension as a process.

VC Dimension for neural network increases with number of parameters and also depends upon non linearity and depth of the network.

Definition 2.2.1 (VC Dimension)**.** *VC Dimension of H is defined by*

$$VC(H) = \max\{m : \Pi_H(m) = 2^m\}.$$

Basically it is the size of the largest set that can be fully shattered by H.

Equivalently, $S \subset U$ where U is a subset of $\mathbf{R^m}$ for some m belongs to natural number. And the concept class C is collection of subsets of U:

$$VC(H) := sup\{cardS : S \text{ shattered by } C\}.$$

Example 2.2.1. *VC(convex d gons) $= 2d + 1$. Consider point on a circle and consider the sequence of alternating sign, then $2d + 2$ points can not be shattered.*

Example 2.2.2. *VC(intervals in R) $= 2$. Any set of two points can be shattered by four points.*

Example 2.2.3. *VC(axis aligned rectangles) $= 4$. Consider a five points configuration like four points are in boundary of a rectangle with same sign and other is inside the rectangle with different sign. So it can not be shattered.*

Example 2.2.4. *$2d+1$ points on a circle can be shattered by a d gon. If $|$positive points$| \leq |$negative points$|$, then polygon inscribed the circle and if $|$positive points$| \geq |$negative points$|$, then circle is inscribed in a polygon.*

Example 2.2.5. *VC(hyperplanes in R^d) $= d + 1$.*

Example 2.2.6. *Consider the parametrized class $F = \{f : f(x) = sign(sin(\theta x)) : \theta \geq 0\}$. Then $VC(F_{sin}) = \infty$, where $X = [0, 2\pi]$.*

Proof. Consider the points $\{(2\pi 10^{-i}, y_i), i = 1(1)n\}$. Now fix $w = \frac{1}{2}(1 + \sum_{i=1}^{n}(\frac{1-y_i}{2}10^i))$. Now $y_j = -1$ and $x_j = 2\pi 10^{-j}$, find wx_j which is equal to $\pi(1 + \epsilon) + 2k\pi$. From $\pi < \pi(1 + \epsilon) < 2\pi$ implies $sin(wx_j) < 0$. It is true for any $n \in$ N. This completes the proof. □

Example 2.2.7. *Now consider 3 points in $2 - D$ euclidean plane. Hypothesis H = set of straight line. Two classes $y = \{1, -1\}$. For every 8 labelling there exit a straight line to classify these 3 points. Therefore 3 points can be shattered. Now consider any 4 points. But this can not be shattered by this H. So $VC(H) = 3$.*

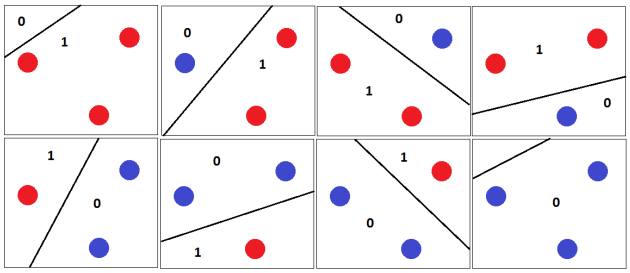


Figure 2.2: Three points shattered by straight lines

2.2.1 VC Definition of Function Classes

The concept of this part has taken from the paper [26]. Let $V \subset \mathbf{R^n}$ for some $n \in \mathrm{N}$. A is a collection of subsets of V which is known as concept class. The concept based definition of VC Dimension is raised in combinatorics and computer science. It is useful to provide an equivalent formulation in terms of functions, which is the way in which the subject arises in statistical estimation.
Instead of A we may define a function class $G = \{g|g : V \to \{0,1\}\}$. To each $g \in G$ we associate the set

$$A_g := \{v \in V : g(v) = 1\}.$$

And thus to G we might associate a concept class,

$$A_G := \{A_g : g \in G\}.$$

we define

$$VC(G) := VC(A_G).$$

Conversely to any concept class A we may formulate a function class G in such a way that $A = A_G$ (just take characteristics function of subsets).

For a set of real valued function class G, we define

$$VC(G) := VC\left(\{H \circ g, g \in G\}\right), where\ H(x) = Heaviside\ function.$$

According to this definition, a subset $W = \{w_1, w_2, ..., w_n\} \subset V$ shattered means, for any combination $e = (e_1, e_2, ..., e_n) \in \{0,1\}^n$, there must exists some function $g = g_e \in G$ which has precisely the same sign, i.e $H(g(u_i)) = e_i$.

2.2.2 Parametric Classes of Functions

The paper [26] describes the following topic in a gentle way. Suppose we have a function

$$\alpha : W \times U \to R$$

where $W = R^\rho$, ρ is number of weights or parameters and consider a parameter vector $w = (w_1, w_2, ..., w_\rho) \in W$. For each choice of parameter vector we get a function:

$$F_\alpha := \{\alpha(w, .) : w \in W\}.$$

and we also define

$$VC(\alpha) := VC(F_\alpha).$$

Example 2.2.8. *Consider the map $\alpha : R^2 \times R \to R$ given by*

$$\alpha((c, d), x) := c \ + \ dx.$$

which imply $U = R$ and C = All open infinite intervals and empty set.

Example 2.2.9. *Consider the function $\alpha : R^3 \times R^2 \to R$ given by*

$$\alpha((c, d, e), (x, y)) := c \ + \ dx \ + \ ey.$$

which also suggests that $U = R^2$ and C = open half spaces.

2.3 Linear Parameterizations

Linear parametrized classes account for vector spaces. And the dimension of this classes is the number of independent parameters. F is a finite dimensional vector space of functions. Dim(F) equal to m iff there exists $\{f_1, f_2, ..., f_m\} \subset F$ such that the following matrix is non singular and any $(m+1)$ by $(m+1)$ matrix of this form is singular.

$$M = \begin{bmatrix} f_1(u_1) & f_1(u_2) & ..... & f_1(u_m) \\ f_2(u_1) & f_2(u_2) & ..... & f_2(u_m) \\ . & . & ..... & . \\ . & . & ..... & . \\ f_m(u_1) & f_m(u_2) & ..... & f_m(u_m) \end{bmatrix}$$

A finite subset $V = \{u_1, u_2, ..., u_m\} \subset U$ is shattered by F if there exit 2^m functions $f_1, f_2, ...f_{2^m}$ such that the matrix formed accordingly to M gives all possible 2^m sign patterns i,e each row vector of the matrix gives one sign pattern.

Lemma 2.3.1 ([26]). *Suppose F is a vector subspace of $\mathbf{R}^U$, then $dim(F) = VC(F)$.*

Proof. Suppose $\{f_1, f_2, ..., f_m\}$ is a linearly independent set and there exit a set $V = \{v_1, v_2, ..., v_m\}$ such that the matrix M has rank m. To show V is shattered it is enough to proof there exit some function $f \in F$ such that $H(f(u_i)) = e_i$ where $e \in \{0, 1\}^m$. M has rank m imply there exit $v \in \mathbf{R^m}$ such that $v^T A = e$. So $f = v^T(f_1, f_2, ..., f_m)$ has given the sign pattern same as e. Therefore $VC(F)$ greater than equal to m. So $dim(F) \leqslant VC(F)$.

Conversely, assume $VC(F)$ is m. So there exit a set $S = \{s_1, s_2, ..., s_m\} \subset U$ is shattered by F. So by definition total 2^m rows of the matrix gives all possible sign pattern. And from this 2^m by m matrix we will get a m by m matrix which is non singular, so dim(F) is at least m. We conclude that $VC(F) \leqslant dim(F)$. □

2.3.1 Affine parameteization

According to paper [26], the concept of this topic describe as follows. Consider $S = \{u_1, u_2, ..., u_m\} \subset U$ is shattered by F and F is a linear space, then the following result holds:
A. For any $e = \{0,1\}^m$, $\delta > 0$, there exists some function $f \in F$ such that $f(u_i) > \delta$ if $e_i = 1$ and $f(u_i) < -\delta$ if $e_i = 0$.

Consider F is a affine subspace, then $G + f_0 = \{g + f_0 : g \in G\}$ where G is a vector subspace of functions and f_0 is a arbitary fixed function. Then $VC(F) = VC(G) = dim(G)$. Enough to show a subset is shattered by G iff it is shattered by F.

If there exists g, h in F such that $H(g(u_i) + f_0(u_i)) = e_i, \forall i$ and $H(h(u_i) + f_0(u_i)) = 1 - e_i, \forall i$. Then $(g-h)(u_i) > 0$ when $e_i = 1$ otherwise $(g-h)(u_i) < 0$. Suppose S is shattered by F then make a function $f = g - h$ where $g, h \in G$ which has the property $H(f(u_i)) = e_i$. For other direction we consider $g \in G$ such that $g(u_i) > \delta$ if $e_i = 1$ and $g(u_i) < -\delta$ if $e_i = 0$, then the result follows.

2.3.2 Perceptron

Here $F =$ all affine function from $\mathbf{R^n}$ to $\mathbf{R}$. So $f(u) = f(u_1, u_2, ..., u_n) = a_0 + a_1u_1 + ... + a_nu_n$.

These functions are linearly parametrized by $(a_0, a_1, ...a_n) \in \mathbf{R^{n+1}}$. So according to Lemma 2.3.1 $VC(F)$ is $n + 1$.

Another approach: Consider $\{u_1, u_2, .., u_n\}$ are inputs and $\{y_1, y_2, ..., y_n\}$ are the corresponding labels. We can define $w_1, w_2, .., w_n$ and bias unit b to ensue $sign(wu_k + b) = y_k$ for all k. Which means

$$sign(wu_k + b) = sign(b + \sum_{j=1}^{n} u_k[j])$$

So we choose $b = 0$ and $w_k = u_k$ for all k, $k = 1(1)n$.

2.4 VC Dimension Related Results

2.4.1 Single Hidden Layer with Fixed Input Weights:

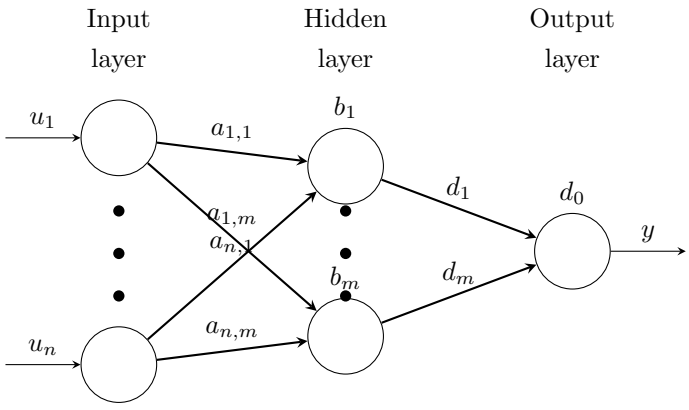


Figure 2.3: Neural net with activation function σ.

The proof of this part has mentioned in [26]. Here M is the m by n input to hidden layer matrix. Assume $d_i's$ are the hidden to output layer weights and $a'_{i,j}s$ for $i = 1(1)m$ and $j = 1(1)n$ are the input to hidden layer weights, $b_1, ..., b_m$ are the hidden layer bias and d_0 is the output layer bias.

$$y = f(u) = d_0 + \sum_{k=1}^{m} d_k.\sigma(M_k.u + b_k)$$

Now weight matrix is $M^T = [a(i,j)]_(n \times m)$ and input vector is $u = (u_1, u_2, ..., u_n)$. And also σ is a arbitary activation function. Consider $M_1, ...M_m$ are the row vectors of M which is fixed and also biases are fixed. Therefore the network is span of 1 and $\sigma(M_i.u + b_i)$ where $i = 1(1)m$. Now by Lemma 2.3.1, we conclude that $dim(F) \leqslant m + 1$.

We can say the bound is tight when σ is "tanh" which follows from the following remark.

Remark 2.4.1. *Assume that $(M_i, b_i) \neq (M_j, b_j)$ for all $i \neq j$ and that $M_i \neq 0$ for all i. $\sigma = tanh$ and consider the remaining as above network. Then $VC(F) = m + 1$.*

If a network have no biases. For an analytic function σ, and for such a net VC dim is n iff σ is not a polynomial.

Lemma 2.4.1 (Sauer)**.** *H be a hypothesis space with $VC(H) = d$, then for all $m \in N$, $\Pi_H(m) \leqslant \sum_{i=0}^{d} \binom{m}{i}$.*

Proposition 2.4.1 ([37])**.** *Let H be hypothesis space with $VC(H) = d$, then $\forall d \leqslant m$,*

$$\Pi_H(m) \leqslant (em/d)^d \leqslant O(m^d).$$

Proof.

$$\begin{aligned}
\sum_{i=0}^{d} \binom{m}{i} &\leqslant \sum_{i=0}^{d} \binom{m}{i} (m/d)^{d-i} \\
&\leqslant \sum_{i=0}^{m} \binom{m}{i} (m/d)^{d-i} \\
&= (m/d)^d \sum_{i=0}^{m} \binom{m}{i} (d/m)^i \\
&= (m/d)^d (1 + (d/m))^m \\
&\leqslant (m/d)^d e^d.
\end{aligned}$$

So either $VC(H) = d < +\infty$ and $\Pi_H(m) = O(m^d)$ or $VC(H) = +\infty$ and $\Pi_H(m) = 2^m$. □

VC of Piece-wise polynomial function: Consider each polynomial is a fixed polynomial. Then the network computes a parametrized polynomial in the input variables, which is a linearly parametrized class (i,e a vector space), follows that the VC is bounded.

2.5 Basic Properties of VC Dimension

Assume F is a set of functions from U to $\{0,1\}$. Then for each n and for all sequence $\{u_1, u_2, ..., u_n\}$, the number of total classification possible for this sequence as a input sequence is:

$$\gamma(u_1, u_2, ..., u_n) := card\{(f(u_1), f(u_2), ..., f(u_n))|f \in F\}$$

So the n element set is shattered iff γ attains its maximum value 2^n. If VC dimension if finite then γ grows polynomially.

For each two non-negative integers $d \leqslant n$, we define $\Phi(n,d)$ is the total number of subsets of a subset has cardinality at most d of a n element set.

$$\Phi(n,d) := \sum_{k=0}^{d} \binom{n}{k} \leqslant 2.\frac{n^d}{d!} \leqslant (en/d)^d$$

Lemma 2.5.1 ([26]). *Let $1 \leqslant n$ and $0 \leqslant d \leqslant n$, and suppose that the matrix $C \in \{0,1\}^{n \times r}$ is so that all its columns are distinct, where r is an integer satisfying $\Phi(n,d) < r$. Then there is some $d+1$ by 2^{d+1} sub matrix of C whose columns are distinct.*

Theorem 2.5.1 (Vapnik-Chervonenkis-Sauer-Shelah). *Suppose $VC(F) = d < \infty$. Then for each $d \leqslant n$ and all sequences $\{u_1, u_2, ..., u_n\}$*

$$\gamma(u_1, u_2, .., u_n) \leqslant \Phi(n,d).$$

[26]. We directly derive from the Lemma 2.5.1. Suppose $VC(F) = d$, then consider any sequence $\{u_1, u_2, ..., u_n\}$, with $d \leqslant n$ and define $k = \gamma(u_1, u_2, .., u_n)$. Now we rearrange all the possible classification as rows in below:

$$M = \begin{bmatrix} f_1(u_1) & f_1(u_2) & ..... & f_1(u_n) \\ f_2(u_1) & f_2(u_2) & ..... & f_2(u_n) \\ . & . & ..... & . \\ . & . & ..... & . \\ f_k(u_1) & f_k(u_2) & ..... & f_k(u_n) \end{bmatrix}$$

If the result does not follows means $\Phi(n,d) < k$. Then from Lemma 2.5.1 there is a sub matrix of order 2^{d+1} by $d+1$. And as each row are distinct so we get a subset of size $d+1$ which is shattered, contradicts our hypothesis. So the result follows. □

Theorem 2.5.2 ([7]). ***A.*** *Consider a parametrized class of binary valued function,*

$$F_f = \{x \to f(x,\theta) : \theta \in R\}, where\ f : \mathrm{R^m} \times \mathrm{R^p} \to \{+1, -1\}.$$

Suppose for each x, $f(x,.)$ can be computed using no more than t operations of the following kinds:

- $+,-,*,/$
- $=,<,>$
- *output* $+1$ *or* -1.

Then $VC(F_f) \leqslant 4p(t+2)$.
B.*Consider all the above operations with one more operation $x \to \exp(x)$. Then $VC(F_f) = O(p^2t^2)$.*

2.5.1 Boolean Closure

One technique to evaluate upper bound on the VC Dimension. Here concept class arises as unions, intersections and other Boolean operations. We have found $F_1, F_2, ..., F_m$ are m classes of functions $U \to \{0,1\}$. And $t : \{0,1\}^m \to \{0,1\}$ is a fixed boolean function. We define:

$$t(F_1, F_2, ..., F_m) := \{t(f_1(.), f_2(.), ..., f_m(.))|f_i \in F_i, i = 1, 2, ..., m\}.$$

Lemma 2.5.2 ([26]). *With $c_m = 2m\log(em)$, a constant which does not depend on the classes F_i nor on the Boolean function t,*

$$VC(t(F_1, F_2, ..., F_m)) \leqslant c_m \max_{i=1,2,...,m}\{VC(F_i)\}.$$

Proof. Suppose $S \subset U$ is shattered and $|S| = n$. Each F_i is a set of functions from S to $\{0,1\}$. Define $F := F_1 \times F_2 \times ..... \times F_m$. So F is a function from S to $\{0,1\}^m$. Since $F \to t(F_1, ..., F_m) : f_1, .., f_m \mapsto t \circ (f_1, .., f_m)$ is onto,

$$card\ t(F_1, ..., F_m) \leqslant card(F) = \Pi_i\ card(F_i).$$

Also $d_i = VC(F_i)$ is finite for all i. Now by Theorem 2.5.1

$$card(F_i) \leqslant (\frac{en}{d_i})^{d_i}, \forall i$$

Let $d = \max_{i=1,2,...,m} d_i$, which gives

$$card\ t(F_1, ..., F_m) \leqslant (\frac{en}{d})^{dm}$$

As S is shattered by F, $2^n \leqslant (\frac{en}{d})^{dm}$. □

Composition Suppose we have $F = \{f|f : U \to V\}$ and $G = \{g|g : V \to W\}$ and we define

$$G \circ F := \{g \circ f|g \in G, f \in F\}$$

a set of function from U to W. We assume given "growth functions" for each class, which bound the number γ of possible classifications, that is, two functions p and q so that

$$\forall S \subset U with\ card\ S \leqslant n, card\ F|_S \leqslant p(n). And\ \forall R \subset V with\ card\ R \leqslant n, card\ G|_R \leqslant q(n).$$

where p *and* q are two growth functions.

Lemma 2.5.3 ([26]). *For each $S \subset U$ with card $S \leqslant n$, card$(G \circ F)|_S \leqslant p(n)q(n)$.*

Proof. We already proved it before. Now we will give a new idea to proof it. Let $S = \{u_1, ..., u_n\}$. Choose a subclass $F_0 = \{f_1, ..., f_{p(n)}\}$ of F such that $F_0|_S = F|_S$. We consider the following subset of V

$$R_i := \{f_i(u_1), ..., f_i(u_n)\}$$

for each $i = 1(1)p(n)$. For each R_i there is $G_i = \{g_1^i, ..., g_{q(n)}^i\}$ of G so that $G_i|_{R_i} = G|_{R_i}$. Now take $g \circ f \in G \circ F$. Consider i and j such that $f|_S = f_i|_S$ and $g|_{R_i} = g^i{}_j|R_i$. Thus $(g \circ f)|_S = (g^i{}_j \circ f_i)|_S$. So we get

$$(G \circ F)|_S = \{g^i{}_j \circ f_i|i = 1(1)p(n), j = 1(1)q(n)\}.$$

This completes the proof. □

2.6 VC Related Results for Multilayer Neural Net

As an application of Lemma 2.5.3, the following result holds for neural network with binary activation functions. This bound is formulated by Cover (1968) and also obtained in Baum and Haussler (1989). Maass (1994) and Sakurai (1993) showed that this bound is tight. Maass's construction follows a network with three layers and binary inputs. Sakurai used two layers and arbitary real numbers as input.

Theorem 2.6.1 ([26]). *The class of functions computed by multilayer neural networks with binary activations and ρ weights has VC dimension $O(\rho \log(\rho))$.*

To finding VC lower bound of two layered neural network with tanh activation functions we use the below theorem. But here all the parameters (weights, bias) and input is restricted to a open subset in real number containing origin.

Theorem 2.6.2 ([22]). *Let F be a class of two layer feedforward neural networks with k hidden units with tanh activation function, input space $X := \{(x_1, ..., x_n) \in \mathrm{R^n} : |x_i| < C\}$, $C > 0$ and $k(n+2)+1$ weights restricted to an open set which include the origin. Then $VC(F_\theta) \geq \mu$ where $\mu = (k-1)(n+1)+1$ is the number of weights of a network with $n-1$ inputs and $k-1$ hidden units.*

Since the network with sigmoid activation function is proportional to a network with tanh activation function as it is a translation of weights, so this result holds for sigmoidal network also. The VC dimension of k-term radial basis function has been shown at least k by Anthony and Holden (1993). When centers are not adjustable, this bound is tight but when the centers are adjustable the below theorem gives a lower bound of VC dimension.

Theorem 2.6.3 ([22]). *Let F be a $k-$term radial basis function with gaussian basis functions. If the input space is $\mathrm{R^n}$, then $VC(F_\theta) \geq \mu$ where $\mu = kn - n$ is the number of parameters in a $k-1$ term radial basis function with $n-1$ inputs.*

The below theorem for VC upper bound of neural network with linear threshold functions does not depends upon depth of the neural network. So according to this theorem the power of a classifier does not depends upon the depth of this classifier (mainly neural network).

Theorem 2.6.4 ([35]). *$F_{p,k}$ be the class of functions computed by a feed forward network of linear threshold functions, with k computation units and p parameters. Then for $p \leqslant n$,*

$$\Pi_{F_{p,k}(n)} \leqslant (enk/p)^p$$

and hence $VC(F_{p,k}) < 2p \log_2(2k/\ln 2)$.

Proof. Fix a set of n input vectors $x_1, x_2, ..., x_n$. Consider topological ordering of the computation units. For computation unit l, let p_l be no of parameters and let $D_l(S)$ be the no of distinct states (i,e parameter settings that compute distinct mappings $\{x_1, x_2, ..., x_n\} \to \{+1, -1\}^l$ from input vector to outputs of computation units up to the $l'th$).

- $D_1(S) \leqslant (en/p_1)^{p_1}$.

- $D_l(S) \leqslant D_{l-1}(S)(en/p_l)^{p_l}$.
- Hence $D_k(S) \leqslant \Pi_{l=1}^{k}(en/p_l)^{p_l}$ and $\log(\Pi_{F_{p,k}(n)}) \leqslant \sum_{l=1}^{k} p_l \log(en/p_l)$.
- bound maximized, $\log(\Pi_{F_{p,k}(n)}) \leqslant p\log(enk/p)$.

This completes the proof. □

VC lower bound for two layered neural network with linear threshold functions follows from the below mention theorem. This bound only depends upon number of computation units and number of weights.

Theorem 2.6.5 ([35]). *$F_{d,k}$ be the class of functions $f : \mathrm{R}^{\mathrm{d}} \to \{+1, -1\}$ computed by a two layer feed forward network of linear threshold functions, with k computation units($p = (d+2)k+1$). Then $VC(F_{d,k}) = \Omega(p)$. A more involved argument shows that $VC(F_{d,k}) = \Omega(p\log(k))$.*

Proof. The idea of proof is given below:

- Arrange kd points in k well separated clusters on the surface of a sphere in R^{d}.
- Ensure d points in each cluster are in general position.
- For each cluster, fit the decision boundary (hyperplane) of a hidden unit to intersect all d points. Oriented so that the unit has output 1 at the centre of the sphere.
- Choose the parameters of the output unit so that it computes the conjunction of its k inputs.
- By perturbing the hidden unit parameters, it is clear that 2^{kd} classification can be computed.

This completes the proof. □

Remark 2.6.1. *Consider the class F of $\{-1, 1\}$ valued functions computed by a network with L layers, p parameters, k computation units with the following non linearities:*

- *Piece-wise constant(linear threshold) = $O(p)$.(Baum,Haussler- 1989)*
- *Piece-wise linear = $O(p^2)$.(Harvey,Liaw,-,-2017)*
- *Piece-wise polynomial = $O(pL^2)$.(Maiorov,Meir - 1998)*
- *Sigmoid = $O(p^2k^2)$.(Karpinsky,Macintyre - 1994)*

2.7 Counting Weights

2.7.1 Multilayer Nets with both H and Linear activation

The number of regions grown by n hyperplanes is a polynomial of n^d, d is the dimension of the space. Let $\Psi(n, d)$ be the largest no of regions into which n hyperplanes can partition $\mathbf{R^d}$. In other words $\Psi(n, d)$ is the best bound for number of connected components created by $H_1, H_2, ..., H_n$ in $\mathbf{R^n}$ of $\mathbf{R^n} - (\bigcup_{i=1}^{n} H_i)$.

The argument has given for neural network with binary activation functions can not be apply to real valued activation functions. Because here the total number of functions on a set of size is n is no more than finite. To overcome these difficulties we follow the below mentioned theorem.

Lemma 2.7.1 ([26]). *For $n \geq d$, $\Psi(n,d) \leqslant \Phi(n,d)$.*

The below Lemma is a construction of a special type of neural network which is used for the proof of Theorem 2.7.4. This result also holds for neural network with skip connections and without skip connections.

Lemma 2.7.2 ([26]). *The network has two H activations and one linear function at a first level, and a linear function at the top level. Suppose that there are a total of g heaviside gates (including one at the top level). Then there exist $r \leqslant g2^{g-1}$ Boolean functions of the form*

$$Q_i(w,u) = H(L_i(w,u)),$$

where each L_i is an affine function of u with parameters w and a boolean function b of r arguments, such that

$$H(\beta(w,u)) = b(Q_1(w,u), Q_2(w,u), ..., Q_r(w,u)), for\ all\ (w,u).$$

The below theorem gives a direct calculation of growth function bound instead of using the concept of vector space dimension for parameters. But this bound only holds for neural network with linear threshold functions.

Theorem 2.7.1 ([35]). *For the class of linear threshold function,*

$$\Pi_{F_d(n)} = 2\sum_{i=0}^{d} \binom{(n-1)}{i}$$

Proof. Fix n points $\{x_1, x_2, ..., x_n\} \in \mathrm{R^d}$. Divide the parameters space of $\{(\theta, \theta_0)\} = \mathrm{R^{d+1}}$ into cells that give the same classification of the points, and count the no of these equivalence classes using a geometric argument.

- Assume the points in S are in general position, i,e all subsets of $\{\binom{x_1}{1}, \binom{x_2}{1}, ..., \binom{x_n}{1}\}$ of size up to $d+1$ are linearly independent. No three in a line, no four are in a plane.
- For each x_i, define the hyperplane $P_i = \{(\theta, \theta_0) \in \mathrm{R^{d+1}} : \theta^T x_i + \theta_0 = 0\}$.
- In order for (θ, θ_0) and $(\theta^1, {\theta_0}^1)$ to label x_i differently, they must lie on opposite sides of P_i(neither on P_i). Thus $|F({x_i}^n)| = CC(\mathrm{R^{d+1}} - \sum_{i=1}^{n} P_i)$, where CC means connected components.
- We define $C(n, d+1) := CC(\mathrm{R^{d+1}} - \sum_{i=1}^{n} P_i)$.
- First $C(1,d) = 2$.
- Next $C(n+1,d) = C(n,d) + C(n,d-1)$. We have n planes in $\mathrm{R^d}$ and add $(n+1)$'th. It splits some of the $C(n,d)$ cells into two, and leaves some of them intact. The number that are split by P_{n+1} is equal to the no of connected components of $P_{n+1} - \sum_{i=1}^{n} P_i$ which is $C(n,d-1)$.
- $C(n,d) = 2\sum_{k=0}^{d-1} \binom{(n-1)}{k}$.

This completes the proof. □

The definition of VC dimension for function class F implies that $VC(F) \leqslant \log_2 |F|$, when $|F|$ is finite. But for some infinite function classes the bound is finite. To reach this argument we follows the below mention two theorems.

Theorem 2.7.2 (Wenocur and Dudley([27])). *$VC(N) = n + 1$ if N consists of a single linear threshold gate with n inputs.*

Corollary 2.7.1. *A linear threshold gate with n inputs can compute at most $|X|^{n+1} + 1$ different functions from any set $X \subset \mathrm{R^n}$ into $\{0, 1\}$.*

As a conjecture we know that the VC dimension can not be greater than the number of parameters. So the below bound can be improved to $O(w)$. Hence with regard to the VC-dimension it is fair to say that a neural net can be "more than the sum of its parts."

Theorem 2.7.3 ([27]). *Let N be an arbitrary feedforward neural net with w weights that consists of linear threshold gates. Then $VC(N) = O(w.\log(w))$.*

Proof. Consider S be the input vector of dimension m of N. By the corollary 2.7.5 a gate g in N can compute at most $|X|^{fan-in(g)+1} + 1$ different functions from any finite set $X \subset \mathrm{R^{fan-in(g)}}$ into $\{0, 1\}$. Hence N can compute at most $\Pi_{g\ gate\ in\ N}(m^{fan-in(g)+1} + 1) \leq m^{2w}$ different functions from S into $\{0, 1\}$. If S is shattered by N then N can compute all 2^m functions from S into $\{0, 1\}$. Then $2^m \leq m^{2w}$ implies $m \leq 2w \log(m)$. From $\log(m) = O(\log(w))$, we get $m = O(w \log(w))$. □

The below theorem is a counterpart of the theorem 2.6.1. This upper bound only depends upon number of weights. This network contains binary as well as linear functions as its activation function.

Theorem 2.7.4 ([26]). *The class of functions computed by multilayer neural networks with binary as well as linear activaton and ρ weights has VC dimension $O(\rho^2)$.*

The below Lemma shows that the upper bound for Theorem 2.7.4 can be attained. This Lemma gives an particular type of architecture to proof the above mentioned claim.

Remark 2.7.1. *The above bound is tight for the given network. Find a family of maps β_ρ where each has $c\rho$ linear and threshold units and each constitute a network architecture. So $VC(\beta_\rho) = \rho^2$ for each ρ. Define*

$$\beta_\rho = \{w \in R | w = \sum_{i=1}^{\rho} b_i/2^i, b_1, b_2, ..., b_\rho \in \{0, 1\}\}.$$

And

$$S_\rho := \{1, 2, ..., \rho\}^2.$$

And also define $\beta_\rho : \mathbf{R}^\rho \times \mathbf{R^2} \to \mathbf{R}$,so that for each $w \in \Lambda_\rho$ and for each $(i, j) \in S_\rho$

$$\beta_\rho((w_1, w_2, ..., w_\rho), (i, j)) = i'th\ bit\ of\ w_j.$$

Chapter 3

VC Dimension of Neural Networks

We will formalize our discussion in somewhat more abstract terms. This chapter will give us different ideas of calculating VC upper bound for different type of neural networks.

3.1 VC Dimension of Piece wise Polynomial Networks

Here the units of neural network contains piece wise polynomial activation functions, specially we will discuss about piece wise linear like ReLU function. First part will give a idea of proof of VC upper and lower bound. The second part of this section will give more tighter bound of the first part of this section.

3.1.1 Linear VC Dimension Bounds for Piece wise Polynomial Network

Goldberg and Jerrum (1995) have shown that the VC upper bound for neural network with piecewise polynomial activation functions is $O(W^2)$. Koiran and Sontag (1997) have demonstrated a network with piecewise polynomial activation functions has VC lower bound $\Omega(W^2)$. But this proof assume a fact that the number of layers can grow with W. But in pratical situation, this number is a small constant. Now the question is whether this bound can be improve?

Theorem 3.1.1 (Upper Bound [1])**.** *Consider a network of real inputs, upto W parameters, upto k computational units arranged in L layers, a single output unit with identity activation function, and all other computation units with piece wise polynomial activation functions of degree l and with p break points, for any positive integer W, $k \leqslant W$, $L \leqslant W$, l and p. F be the class of real valued functions computed by this network. Then*

$$VC(sgn(F)) \leqslant 2WL\log(2eWLpk) + 2WL^2\log(l+1) + 2L$$

and also if p, l are fixed and since L,k are $O(W)$, implies that

$$VC(sgn(F)) = O(WL\log L + WL^2)$$

Proof. The below part will give us a idea of the whole proof which has discussed in the main paper. Basically,

this bound holds for piece wise polynomial neural network.

- Fixed x as an input, output of the network is $f(x, a)$ corresponds to a piece wise polynomial of parameter a and degree of this polynomial no more than $(l+1)^{L-1}$.
- Parameter domain $A = \mathrm{R}^{\mathrm{W}}$ can be split into regions, in each of which the function $f(x, .)$ is a polynomial.
- To obtain an upper bound of the number of sign assignments, that can be attained by varying the parameters of a set of polynomials.
- $x_1, x_2, ..., x_m$ are m arbitary points.
- Target is bound $K = |\{(sgn(f(x_1, a)), ..., sgn(f(x_m, a))) : a \in A\}|$.
- If we consider a partition $S_1, S_2, ..., S_N$ of parameter domain A.
- Then $K \leqslant \sum_{i=1}^{N} |\{(sgn(f(x_1, a))..., sgn(f(x_m, a))) : a \in S_i\}|$.
- Choose partition such that within each region $f(x_1, .), ..., f(x_m, .)$ are all fixed polynomials of degree no more than $(l+1)^{L-1}$.
- Then each term of the summation less than equal to $2((2em(l+1)^{L-1})/W)^W$.
- Construct the partition and determine an upper bound of its size.
- Let S_1 be a partition of A such that, for all $S \in S_1$, there exists constants $b_{h,i,j} \in \{0, 1\}$ for which $sgn(p_{h,x_j}(a) - t_i) = b_{h,i,j}, \forall a \in S$, where $j \in [m], h \in [k_1], i \in [p]$.
- t_i are the break points of the piece wise polynomial functions and $p_{h,x_j} = a_h.x_j + a_{h,0}$ where $a_h \in \mathrm{R}^{\mathrm{d}}, a_{h,0} \in \mathrm{R}$ are the weights of the h'th unit in the first layer.
- S_1 is determined by only parameters of the first hidden layer.
- Clearly, for $a \in S$, the output of any first layer unit in response to an x_j is a fixed polynomial in a.
- Let $W_1, ..., W_L = W$ be the number of variables used in computing the unit outputs up to layer $1, 2, ..., L$ respectively, and $k_1, .., k_L = 1$ be the number of computation units in layer $1, 2, ..., L$ respectively.
- Choose S_1 so that $|S_1|$ is no more than the number of sign assignments possible with $mk_1.p$ affine functions in W_1 variables.
- Then $|S_1| \leqslant 2((2empk_1)/W_1)^{W_1}$.
- Assume for all $S \in S_{n-1}$ and all x_j, the net input of every unit in layer n in response to x_j is a fixed polynomial of $a \in S$, of degree no more than $(l+1)^{n-1}$.
- Let S_n be a partition of A that is refinement of S_{n-1}, such that for all $S \in S_n$ there exists constants $b_{h,i,j} \in \{0, 1\}$ such that $sgn(p_{h,x_j}(a) - t_i) = b_{h,i,j}$ for all $a \in S$. p_{h,x_j} is describing the net input of the h'th unit in the n'th layer, in response to x_j.
- Refinement: For all $S \in S_n$, there exist an $S' \in S_{n-1}$ such that $S \subset S'$.

- Output of each n'th layer unit is a fixed polynomial in a of degree no more than $l(l+1)^{n-1}$ for all $a \in S$.
- Choose S_n such that for all $S' \in S_{n-1}$ we have $|\{S \in S_n : S \subset S'\}|$ is no more than the number of sign assignments of mpk_n polynomials in W_n variables of degree no more than $(l+1)^{n-1}$.
- So this is no more than $2((2empk_n(l+1)^{n-1})/W_n)_n^W$.
- Net input of every unit in layer $n+1$ is a fixed polynomial of $a \in S \in S_n$ of degree no more than $(l+1)^n$.
- Finally S_{L-1} of A such that for all $S \in S_{n-1}$, the network output is a fixed polynomial of $a \in S$ of degree no more than $l(l+1)^{L-2}$ in response to x_j.
- $|S_{L-1}| \leqslant 2((2empk_1)/W_1)^{W_1}.\Pi_{i=2}^{L-1}2((2empk_i(l+1)^{i-1})/W_i)^{W_i}$.
- $K \leqslant \Pi_{i=1}^{L}2((2empk_ip(l+1)^{i-1})/W_i)^{W_i}$.
- $m < L + \sum_{i=1}^{L} W_i \log((2empk_i(l+1)^{i-1})/W_i)$, log is base 2.

This completes the proof. □

Theorem 3.1.1 gives an upper bound $O(WL^2 + WL\log(WL))$. If L os fixed this is $W\log(W)$ which is better than $O(W^2)$. The below theorem gives a lower bound $\Omega(WL)$, $L = O(W)$. This generalize the result of Koiran and Sontag as it holds for any number of layers. But this lower bound holds for neural network with continuous activation functions.

Theorem 3.1.2 (Lower Bound [1]). *$f : \mathrm{R} \to \mathrm{R}$ is a function with following properties:*

A. *$lim_{\alpha\to\infty}f(\alpha) = 1$ and $lim_{\alpha\to-\infty}f(\alpha) = 0$.*

B. *f differentiable in some point x_0 with $f'(x_0)$ not equal to zero.*

Now for any $L \geq 1$ and $W \geq 10L-14$, there exists a feed forward neural network with following properties: The network has L layers, W parameters, output unit has a linear function and all other units have f as a activation functions. Then the set $sgn(F)$ of functions computed by the network has

$$VC(sgn(F)) \geq (L/2)(W/2).$$

W = number of parameters or edges. L = number of layers.

Remark 3.1.1 (Upper Bound). *Here we organized some VC upper bounds.*

- $O(WL\log W + WL^2)$*[[1]].*
- $O(W^2)$*[[7]].*
- $O(W\log W)$*[[34]], NN with linear threshold function.*

Remark 3.1.2 (Lower Bound). *This remark contains some VC lower bounds.*

- $\Omega(WL)$*[[1]]*
- $\Omega(W\log W)$*[M'98].*
- $\Omega(W\log W)$*[[18]], NN with linear threshold.*

- $\Omega(WL)$*[Bartlett'98], NN with linear threshold and identity function.*

The below theorem gives a upper bound on VC dimension for neural network with ReLU activation functions. This bound tight for any parameter range. All of these bound generalize for arbitary piecewise linear class of activation functions. Theorem 3.1.4 gives a proof idea for lower bound and Theorem 3.3.5 gives a proof idea for upper bound.

3.1.2 Refinement of [1]

Theorem 3.1.3 ([17]). *For a ReLU neural network W params, L layers $\Omega(WL\log(W/L)) \leqslant VCDim \leqslant O(WL\log W)$.*

Theorem 3.1.4 ([16]). *Lower Bound(Refinement of [1])*

Proof. Here instead of one bit per layer, we are taken multiple bits.

- Shattered set $S = \{e_i\}_{i\in[n]} \times \{e_j\}_{j\in[m]}$.
- Encode f with weights $a_i = 0.a_{i,1}, a_{i,2}, ..., a_{i,m}$ where $a_{i,j} = f(e_i, e_j)$.
- Given e_i, easy to extract a_i.
- Design bit extractor to extract $a_{i,j}$. One bit per layer imply $\Omega(WL)$ and $\log(W/L)$ bits per layer imply $\Omega(WL\log(W/L)$.

This completes the proof □

Theorem 3.1.5 ([16]). *Upper bound(Refinement of [1])*

Proof. Here concept of depth upto layer $i'th$ is added, which is a function of weights of the predefined neural network.

- Fix a shattered set $X = \{x^1, x^2, ..., x^m\}$.
- Partition parameter space such that input to 1'st hidden layer has constant sign. Can replace σ with 0 (if < 0) or identity (if > 0)!
- Size of partition is small, i,e $\leqslant (cm)^W$ by Warren'68.
- Repeat procedure for each layer to get partition of size $\leqslant (cLm)^{O(WL)}$.
- In each piece, output is polynomial of deg L. So total no of signing $\leqslant (cLm)^{O(WL)}$.
- Since X is shattered, $2^m \leqslant (cLm)^{O(WL)}$ which imply $m = O(WL\log W)$.

This completes the proof. □

The below theorem follows bit extraction technique. Here instead of one bit extraction they extract many bits per iteration time. As a result the new obtained bound is tighter than the previous calculated bound.

Lemma 3.1.1 ([16]). *Suppose a ReLU neural network of W params, L layers extract $m'th$ bit of input. Then $m \leqslant O(L\log(W/L))$.*

The below theorem gives a lower bound on VC dimension for neural network with ReLU activation functions. Goldberg and Jerrum (1995) has been given a lower bound on VC dimension for neural network with piecewise polynomial activation functions. Here the authors used the fact that a function that can be expressed as a Boolean formula containing s distinct atomic predicates.

Lemma 3.1.2 ([16]). *There exit constant c such that the following holds. Given any W, L with $C^2 < CL < W$, there exists a ReLU network with $\leqslant L$ layers and $\leqslant W$ parameters with VC Dimension $\geq WL\log_2(W/L)/c$.*

The below theorem imply that the bit extraction approach cannot give a lower bound better than the calculated bound.

Lemma 3.1.3 ([16]). *Assume there exists a neural network with W parameters, L layers that computes a function $f : \mathrm{R} \to \mathrm{R}$, with the property that $|f(x) - (x \bmod 2)| < 1/2$ for all $x \in \{0, 1, ..., 2^m - 1\}$. Also suppose the activation functions are piece wise polynomial of degree at most $d \geq 1$ in each piece, and have at most $p \geq 1$ pieces. Then we have $m \leq L\log_2(13pd^{(L+1)/2}W/L)$. For Piece wise linear this gives $m = O(L\log(W/L))$.*

The below theorem applicable for neural network with piecewise linear activation functions. And the input of this neural network is arbitary real domain.

Lemma 3.1.4 ([16]). *Consider piece-wise linear neural network with W parameters arranged in L layers. Let F be the set of real valued function computed by this network. Then if $m = VC(sgn(F))$ and p is no of pieces of the activation function, it holds that $m \leqslant 4W(L+1)\log_2(2eWmp)$. So $VC(sgn(F)) = O(WL\log W)$.*

Proof. For piece-wise polynomial we know $O(W^2)$ by ([13]) and $O(WL^2 + WL\log W)$ by (Bartlett'98). This proof is similar to (Bartlett'98). Here use a result [warren(1968)]. □

The below theorem gives upper bound on VC dimension for neural network with piecewise polynomial activation functions. Also this network supports both with skip connections and without skip connections. Here the authors imposed a new term L_1. The new modified bound depends upon this term.

Theorem 3.1.6 ([16]). *Consider a neural network with W parameters, U computational units arranged in L layers, so that each unit has connection only from units in earlier layers.Let k_i denote the number of units at the $i'th$ layer.Suppose that all non output units have piece wise polynomial activation functions with $p+1$ pieces and degree no more than d, and the output unit has identity function as its activation function.*

If $d = 0$, let W_i denote the number of parameters(weights and biases) at the inputs to units in layer i; if $d_i > 0$,let W_i denote the total number of parameters(weights and biases) at the inputs to units in all the layers up to layer $i(1, 2, ...i)$.Define the effective depth as

$$L_1 := 1/W \sum_{i=1}^{L} W_i,$$

and let

$$R := \sum_{i=1}^{L} k_i(1 + (i-1)d^{i-1}) \leqslant U + U(L-1)d^{L-1}.$$

Case I *For the class F of all(real valued) functions computed by this network and $L_1W \leqslant m$,we have*

$$\Pi_{sgn(F)}(m) \leqslant \Pi_{i=1}^{L} 2((2empk_i(1+(i-1)d^{i-1}))/W_i)^{W_i}$$

Case II *If $U > 2$,then*

$$VC(F) \leqslant L + L_1W\log(4epR\log(2epR)) = O(L_1W\log(pU) + LL_1W\log d).$$

Case III *If $d = 0$, then*

$$VC(F) \leqslant L + W\log(4epU\log(2epU)) = O(W\log(pU)).$$

Case IV *If $d = 1$,then*

$$VC(F) \leqslant L + L_1W\log(4ep\sum(ik_i\log(\sum(2epik_i)))) = O(L_1W\log(pU)).$$

3.2 VC Dimension of Sigmoidal Neural Networks

In this section we discussed about neural network with sigmoid and radial basis functions. We focused on VC finiteness of the network and also give an idea of VC for definable sets.

3.2.1 Lower Bound of Sigmoidal Network which Approximate Continuous Functions

Definition 3.2.1. *A dichotomy of a set $S \subset \mathrm{R^n}$ is a partition of S into 2 disjoint subsets S_0, S_1 such that $S_0 \cup S_1 = S$.*
According to functions: *For a set of functions F mapping from $\mathrm{R^n}$ to $\{0,1\}$ and a dichotomy S_0, S_1 of S, we say F induced the dichotomy if there is a $f \in F$ such that $f(S_0) \subset \{0\}$, $f(S_1) \subset \{1\}$.*

Definition 3.2.2. *F shatters S if F induced all dichotomies on S.*

Basics

We show how to construct a neural network N_n that computes some of the polynomials. This architecture has only one programmable parameter. Let the sequence of polynomials over R is defined by $p_n(x) = 4x(1-x)$ when $n = 1$ and $p_n(x) = p(p_{n-1}(x))$ when $n \geq 2$. So p_n has degree 2^n.

Theorem 3.2.1 ([3]). *The polynomial (p_n) approximated in $[0,1]$ by a sigmoidal neural network with error rate $O(2^{-n})$ in the l_∞ norm must have at least $\Omega(n^{1/4})$ computation nodes.*

Proof. Here we will give a basic idea about how the original proof is going.

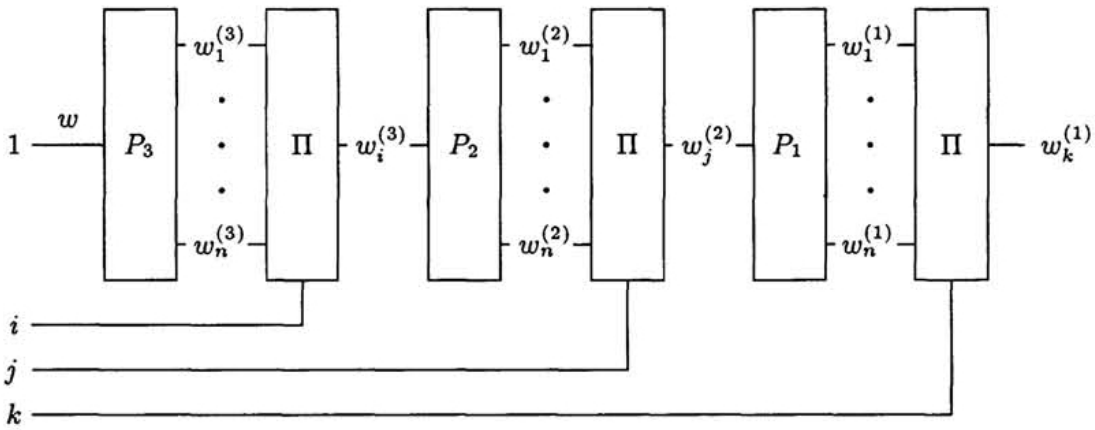


Figure 3.1: The Network N_n

- Input values of input nodes are $x_4 = 1$, $x_3 = i$, $x_2 = j$, $x_1 = k$.
- Only programmable weight w is associated with outgoing edge from node x_4.
- Computation nodes are divides into six labels, each label is a network.
- Three labels, denoted by Π having $n + 1$ input nodes and 1 output node.
- Each calculate projections $\pi : \mathrm{R}^{n+1} \to \mathrm{R}$ where $\pi(y_1, ..., y_n, a) = y_a \ for \ a \in [n]$.
- The levels P_1, P_2, P_3 have n output nodes and one input node each. P_3 receives 1 as a input.
- Define output of P_λ for $\lambda \in [3]$ by ${w_b}^\lambda = p_{b.n^{\lambda-1}}(v), b \in [n]$ where v denotes input value of level P_λ.
- This value equal to w for $\lambda = 3$ and $\pi({w_1}^{\lambda+1}, ..., {w_n}^{\lambda+1}, x_{\lambda+1})$ otherwise.
- Also ${w^\lambda}_{b+1}$ can be calculated from ${w_b}^\lambda$ as $p_{n^{\lambda-1}}({w_b}^\lambda)$.
- Therefore computation of level P_λ contains n gates, each of them computing the function $p_{n^{\lambda-1}}$.
- To show N_n can shatter a set of cardinality n^3.

Rest of the proof and details of the proof is in [3]. □

3.2.2 Polynomial Bound for VC Dimension of Sigmoidal Networks

In this section the VC upper bound of sigmoidal neural network is calculated using model theory approach. Basically they first tried to compute VC upper bound of definable sets and then tried to establish a representation of this neural network using terms, o-minimal set.

Definition 3.2.3 (Term)**.** *A term defined as follows*

- *A variable is a term.*
- *A constant symbol is a term.*

- *If F is a m-placed function symbol and $t_1, .., t_m$ are terms, then $F(t_1, .., t_m)$ is a term.*
- *A string of symbols is a term iff it can be shown to be a term by finite number of application of above three steps.*

Definition 3.2.4 (Formula)**.** *A formula defined as follows*

- *If t_1, t_2 are terms, then $(t_1 = t_2)$ is a formula.*
- *If R is a n-placed relation symbol and $t_1, .., t_m$ are terms then $R(t_1, ..., t_m)$ is a formula.*
- *If ϕ is formula, then $\neg\phi$ is a formula.*
- *If ϕ, χ are formulas, then $\phi \vee \chi$, $\phi \wedge \chi$, $\phi \rightarrow \chi$, $\phi \iff \chi$ are also formulas.*

Definition 3.2.5 (Language)**.** *A Language L is*

- *A set F of function symbols and a positive integer n_f for each $f \in F$.*
- *A set R of relation symbols and a positive integer n_R for each $r \in R$.*
- *A set C of constant symbols.*

Example 3.2.1. *Language of Rings: $F = \{+, -, .\}$, $n_+ = n_. = n_- = 2$,$R = \phi, C = \{0, 1\}$.*
Language of group: $F = \{.\}$,$n_. = 2$,$C = \{1\}$,$R = \phi$.

Definition 3.2.6. *An L structure $\mathcal{M}$ is*

- *A non empty set M, underlying set of the structure.*
- *A function $f^{\mathcal{M}} : M^{n_f} \rightarrow M$ for each $f \in F$.*
- *A relation $R^M \subset M^{n_R}$ for each $R \in \mathcal{R}$.*
- *An element $C^{\mathcal{M}} \in M$ for each constant c.*

Each first order structure $\mathcal{M}$ has a satisfaction relation $M \models \phi$ defined for all formulas ϕ in the language consisting of the language of $\mathcal{M}$ together with a constant symbol for each element of M.

A structure $\mathcal{M}$ is said to be a model of a theory T if the language of $\mathcal{M}$ is the same as the language of T and every sentence in T is satisfied by $\mathcal{M}$.

Definition 3.2.7 (Definable Relation)**.** *An n-ary relation R on the universe M of a structure $\mathcal{M}$ is said to be definable if there exists a formula $\phi(x_1, x_2, ..., x_n)$ such that $R = \{(a_1, ..., a_n) \in M^n : M \models \phi(a_1, ..., a_n)\}$. There exists a ϕ such that $(a_1, ..., a_n) \in R$ iff $\mathcal{M} \models \phi(a_1, .., a_n)$ is correct.*

Definition 3.2.8. *M is $o-$minimal if for every formula $\Phi(v_1, .., v_l)$ and every $\beta \in \mathrm{M}^{\mathrm{l}}$, Φ_β is a finite union of intervals with endpoints in $M \cup \{\infty, -\infty\}$.*

Model Theoretic Preliminaries

The concept of this part has taken from [9]. We consider a Network A with activation function σ and also have k inputs, m computation units, l weights. It has output in $\{0,1\}$ set. By model theory we can express A by a exponential formula $\Phi(v,y)$, where $v \in \mathrm{R^k}$ and $y \in \mathrm{R^l}$, which is a combination of polynomials, activation functions over the computation nodes of A. The functions computed by A represent by $\Phi(v,y) > 0$.

In other way we can say A could be express a Boolean combination of atomic formulas of two forms $\tau(v,y) = 0$ or $\tau(v,y) > 0$, describing local computations of A at its computation nodes. Now VC of A is the VC of the class $C_\Phi = \{\Phi_\beta : \beta \in \mathrm{R^l}\}$ for $\Phi_\beta = \{x \in \mathrm{R^k} : \Phi(x,\beta) > 0\}$, the partition of $\mathrm{R^k}$ by A according to the weight assignment β.

We turn our attention to the analysis of general formula resulting from the local computations. We consider structure M on real field consists of C^∞ functions. L is a first order language consists primitives $<, 0, 1, -, ., +$, together with n ary function symbols f. Each f has a fixed interpretation by a C^∞ function $f' : \mathrm{R^n} \to \mathrm{R}$, thereby determining an L structure M.

If $\tau(v_1, V_2, .., V_M)$ is an L term with free variables $v_1, v_2, ..., v_m$, τ defines a m ary C^∞ function from $\mathrm{R^m}$ to R. L formulas $\Phi(v_1, v_2, ..., v_k)$ defines subset of $\mathrm{R^k}$, and L formulas $\Phi(v_1, v_2, .., v_k, y_1, .., y_l)$ together with $\beta = (\beta_1, ..., \beta_l) \in \mathrm{R^l}$ defines subset of $\mathrm{R^k}$, namely $\Phi_\beta = \{x \in \mathrm{R^k} : M \models \Phi(x,\beta)\}$. For $\Phi(v,y)$ as above, let $\{\Phi_\beta : \beta \in \mathrm{R^l}\} = C_\Phi$. Also C_Φ is a definable family of definable sets.We calculate VC dimension of C_Φ.

Theorem 3.2.2 ([9]). *VC dim of* $\Phi \leqslant 2\log B + (17\log s)l$.

Application to Sigmoidal Networks

The fundamental idea behind this proof has taken from [9]. Consider a sigmoidal network A with l weights $y_1, ..., y_l$ or programmable parameters, k input nodes $v_1, ..., v_k$ and one output node. The m'th computation node known as N_m is labelled by a variable z_m, and a polynomial $P_m(v_{t_1}, ..., v_{t_\rho}, z_{u_1}, ..., z_{u_\gamma}, y_{\lambda_1}, ..., y_{\lambda_\delta})$, where $y's$ are subset of the weight variables and $v's$ correspond to the input nodes immediately below m (i,e. connected to m) and $z's$ correspond to the computation nodes immediately below m.

Then A computes a function $\beta_A : \mathrm{R^{k+l}} \to \mathrm{R}$. If N is a computation node as above labelled by z_m, then $f_N(v,y) = P_m(v_{t_1}, ...v_{t_\rho}, \sigma(f_{N_1}(v,y)), ..., \sigma(f_{N_\gamma}(v,y)), y_{\lambda_1}, ..., y_{\lambda_\delta})$, where N_i corresponds to u_i for $1 \leqslant i \leqslant \gamma$. Also β_A is f_{N_w} where N_w is the output node.

For the case of a language with $+, -, ., 0, 1$ and a symbol σ for a activation function, the $f_A(v,y)$ is given by the term $\tau(v,y)$, by transcribing naively the above recursion. Assume $\Phi(v,y)$ be $\tau(v,y) > 0$. The VC dimension if finite since σ is definable in $+, -, ., 1, 0, \exp x$.

We have simply to bound $\Gamma'(\tau, j), j \leqslant l$ in order to get B. For this we should calculate number of connected components of an intersection of no more than j sets of the form $\{y : \tau(\alpha_i, y) = \epsilon_i\}, 1 \leqslant i \leqslant j$. This estimate will get from Khovanski estimate.

Z_m is use as a computation variable and Z'_m is those correspondence with this. Z_w consider as a output variable. Now consider $\sum_m [(Z_m - P_m(v_{t_1}, ..., v_{t_\rho}, Z'_{N_1}, ..., Z'_{N_\gamma}, y_{\lambda_1}, ..., y_{\lambda_\gamma}))^2 + (1 - Z`_m(1 + \exp^{-Z_m}))^2] = \mu(v,z,y)$. Notice that $\mu(v,z,y) = 0 \to Z_w = \tau(v,y)$ *and* $Z_w = \tau(v,y) \iff (\exists z)\mu(v,z,y) = 0$. By [8] for fixed α the number of connected components in $\mathrm{R^{l+2m}}$ of $\mu(\alpha,z,y) = 0$ is $\leqslant 2^{m(m-1)/2}(2d)^{l+2m}[(l+2m+1)(2d+1)]^{l+3m}$, where m is the number of computation units.

From here we will get a bound for $\tau(\alpha,y) = \epsilon$. But we need to handle $\leqslant j$, $\tau(\alpha_i,y) = \epsilon_i$ together. Now we need $v_{r,i}, Z_{N,i}, Z'_{N,i}, i \leqslant j$ as variables. We obtain an estimate $2^{mj(mj-1)/2}(2d)^{l+2mj}[(l+2mj+1)(2d+1)]^{l+3mj}$ in $\mathrm{R^{l+2mj}}$ space. Since B can be chosen no larger than the supremum of these $j \leqslant l$, we get

$\log(B) \leqslant (ml)(ml-1)/2 + (l+2mj)(\log(2d)) + (l+3ml)\log(l+2ml+1) + (l+3ml)\log(2d+1)$. So VC(A) $\leqslant (ml)(ml-1)/2 + l(1+2m)(\log(2d)) + l(1+3m)\log(l(2m+1)+1) + l(1+3m)\log(2d+1)$.

3.2.3 Additional Activation Functions

Definition 3.2.9. *A two layer sigmoid network with n inputs, W weights, and a single real valued output is described by the function* $f_S : \mathrm{R}^{\mathrm{W}} \times X \to \mathrm{R}$, *where* $X \subset \mathrm{R}^{\mathrm{n}}$, $f_S(\theta, x) = a_0 + \sum_{i=1}^{k} \frac{a_i}{1+e^{-(b_i.x+b_{i0})}}$, *with* $a_i \in \mathrm{R}$, $b_i = (b_{i1}, ..., b_{in}) \in \mathrm{R}^{\mathrm{n}}$ *and* $\theta = (a_0, ..., a_k, b_{10}, ..., b_{kn}) \in \mathrm{R}^{\mathrm{W}}$. *In this case* $W = kn + 2k + 1$. *A RBF network is described by the function* $f_{RBF}(\theta, x) = a_0 + \sum_{i=1}^{k} a_i.e^{-||x-c_i||^2}$, *where* $c_i = (c_{i1}, ..., c_{in}) \in \mathrm{R}^{\mathrm{n}}$ *and* $\theta = (a_0, .., a_k, c_{11}, ..., c_{kn}) \in \mathrm{R}^{\mathrm{W}}$. *Here* $W = kn + k + 1$.

Theorem 3.2.3 ([22]). *Let* $X = \{-D, ..., D\}^n$ *for some positive integer D. For the sigmoid and RBF networks,* $f_S, f_{RBF} : \mathrm{R}^{\mathrm{W}} \times X \to \mathrm{R}$, *we have*

$$VC(f_S) \leqslant 2W \log_2(24eWD)$$

$$VC(f_{RBF}) \leqslant 4W \log_2(24eWD)$$

Proof. For any $\theta \in \Theta, x \in X, r \in \mathrm{R}$, let

$$f_{S'}(\theta, (x, r)) = (f_S(\theta, x) - r)(\Pi_{i=1}^{k}\Pi_{j=1}^{n} e^{-b_{ij}D})(\Pi_{i=1}^{k}(1 + e^{-(b_i.x-b_{i0})}))$$

Clearly, $f_{S'}(\theta, (x, r))$ always has the same sign as $f_S(\theta, x) - r$, since the denominators in $f_S(\theta, x)$ is always positive. So $dim(f_S) \leqslant VC(f_{S'})$. But $f_{S'}(\theta, (x, r))$ is polynomial in $\theta' = (a_0, .., a_k, e^{-b_{10}}, ..., e^{-b_{kn}})$, with degree no more than $2Dnk + k + Dn + 1 < 3WD$. From [7] we get $VC(f_{S'}) < 2W \log_2(24eWD)$. □

3.3 Bounding The VC Dimension of Concept Class Parametrized by Real Numbers

Here we establish bound on the VC dimension of non discrete concept classes. Assume X is an instance space.

Definition 3.3.1. *The membership test of a concept class* $\mathcal{C}$ *over domain X takes as input a concept* $C \in \mathcal{C}$ *and instances* $a \in X$, *and returns the boolean value* $a \in C$.

The membership test for $\mathcal{C}_{k,n}$ as defined above, is assumed to be expressed as a formula $\Phi_{k,n}$ (in the first order theory of the reals) with $k + n$ free variables representing a concept C and instance a. Or as an algorithm $\mathcal{A}_{k,n}$, similarly taking $k + n$ real inputs, which uses exact real arithmetic and returns the truth value $a \in C$. We say $\mathcal{C}_{k,n}$ is defined by $\Phi_{k,n}$ or $\mathcal{A}_{k,n}$.

Example 3.3.1. $k = m(n + 1)$ *where m is a positive integer. Define* $\Phi_{k,n} = \bigcup_{i=1}^{m}[\sum_{j=1}^{n}(x_j - a_{ij})^2 \leqslant {r_i}^2]$. *It defines the concept class whose elements are unions of m balls in n dimensional Euclidean space.* a_{ij} *parameterize the centers of the m balls,* r_i *be their radii,* x_j *is cartesian co ordinates of the instance.*

We focus on the result of [10] which exhibits a NASC on a first order formula over some structure to define a class of finite VC dimension.

3.3.1 Upper Bounds

A concept C and instance a will be represented by the sequence of reals $(y_1, .., y_k)$ and $(x_1, .., x_n)$ respectively. A sign assignment to polynomial p is one of the (in)equalities $p > 0$, $p = 0$, $p < 0$, a sign assignment to a set of m polynomials is consistent if all m equalities can be satisfied by some assignment of real numbers to the variables. A non zero sign assignment is one which has no equalities.

Theorem 3.3.1. *Warren Result given in [11].*

Corollary 3.3.1 ([13]). *If $p_1, ..., p_m$ is a set of polynomials of degree at most $d \geq 1$ in n real variables with $m \geq n$, then the number of consistent non zero sign assignments to the p_i is at most $(8edm/n)^n$.*

Proof. Let $\mathcal{P} = \{p_1.., p_m\}$. Consider the set of polynomials $\mathcal{P}_1 = \{p_1 + \epsilon, p_1 - \epsilon, ..., p_m + \epsilon, p_m - \epsilon\}$. To proof for $\epsilon > 0$, every sign assignments to $\mathcal{P}$ corresponds to a unique non zero sign assignment to $\mathcal{P}_1$.

Milnor theorem gives an upper bound on the number of connected components of the subset of R^n corresponding to any sign assignment. The size of a formula refers to the number of distinct atomic predicates that it contains. □

For polynomial learn ability we need upper bound on VC dimension that is polynomial in synthetic complexity of concepts. This result trivial for discrete input cases but non trivial for generalized case of examples and concepts. The author has been showed that this result true for two generalized classes. One is classes where the criterion for membership of an instance in a concept can be expressed as a formula (in the first order theory of reals) with fixed quantification depth and exponentially bounded length, whose atomic predicates are polynomial inequalities of exponentially bounded degree.

Theorem 3.3.2 ([13]). *Let $\{\mathcal{C}_{k,n} : k, n \in \mathrm{N}\}$ be a family of concept classes where concepts in $\mathcal{C}_{k,n}$ and instances are represented by k and n real numbers, respectively. Suppose that the membership test for any instances a in any concept C of $\mathcal{C}_{k,n}$ can be expressed as a boolean formula $\Phi_{k,n}$ containing $s = s(k, n)$ distinct atomic predicates, each predicate being a polynomial inequality or equality over $k + n$ variables (representing C and a) of degree at most $d = d(k, n)$. Then $VC(\mathcal{C}_{k,n}) \leqslant 2k \log(8eds)$.*

Corollary 3.3.2. *If the size s and degree d are both at most exponential in k and n, then the VC dimension of $\mathcal{C}_{k,n}$ is polynomially bounded in k, n.*

The other is classes where containment of an in a concept is testable in polynomial time, assuming we may compute standard arithmetic operations on reals exactly in constant time.

Theorem 3.3.3 ([7]). *Let $\{\mathcal{C}_{k,n} : k, n \in \mathrm{N}$ be a set of concept classes as before, for which the test for membership of an instance a in a concept C consists of an algorithm $\mathcal{A}_{k,n}$ taking $k+n$ real inputs representing C and a, whose run time is $t = t(k, n)$, and which returns the truth value $a \in C$. The algorithm $\mathcal{A}_{k,n}$ is allowed to perform conditional jumps and execute the standard arithmetic operations on real numbers $+, -, ., /$ in constant time. Then $VC(\mathcal{C}_{k,n}) = O(kt)$.*

Corollary 3.3.3. *Let $\mathcal{C}_{k,n}$ be as in the above theorem. If the run time of algorithm $\mathcal{A}_{k,n}$ is polynomially bounded in k and n, then so is the VC dimension of the concept class $\mathcal{C}_{k,n}$.*

There exists an algorithm with runtime t, defines a concept class of VC dimension $\Omega(kt)$.

Theorem 3.3.4. *See [12]. It is a quantifier elimination procedure to give us a quantifier free formula of the original form.*

Corollary 3.3.4 ([13]). *Let $\{\mathcal{C}_{k,n} : k, n \in \mathrm{N}\}$ be a set of concept classes as before. Suppose that the membership test of a given instance a in a given concept C can be expressed as a formula $\chi_{k,n}$ in the first order theory of the real numbers with $k+n$ free variables representing C, a; suppose further that the number of bound variables in polynomial in k, n, that the depth of alteration of quantifiers is uniformly bounded, and that the atomic predicates are bounded in number and degree by an exponential function of k, n. Then the VC dimension of $\mathcal{C}_{k,n}$ is polynomial in k, n.*

3.4 VC Dimension from Geometric Approach

For calculating VC dimension of linear threshold gates, the proof involve the number of distinct output of all linear units along with input varies on m patterns. But the sigmoidal network have infinitely many output, so this technique does not work. To overcome this issues, this section has come in computational learning theory field.

3.4.1 Some Notions of Geometric Approach

Definition 3.4.1. *Let H be a class of $\{0,1\}$ valued functions defined on X, and F be a real valued function defined on $\mathrm{R}^{\mathrm{d}} \times X$. We say H is a $k-$combination of $sgn(F)$ if there is a Boolean function $g : \{0,1\}^k \to \{0,1\}$ and functions $f_1, f_2, ..., f_k$ in F so that for all $h \in H$ there is a parameter $a \in \mathrm{R}^{\mathrm{d}}$ such that $h(x) = g(sgn(f_1(a,x)), ..., sgn(f_k(a,x)))$ for all $x \in X$.*

Definition 3.4.2. *A set $\{f_1, f_2, ..., f_k\}$ of differential functions mapping from R^{d} to R is said to have regular zero set intersection if for all non empty subset $\{i_1, i_2, ...i_l\} \subset \{1, 2, ..., k\}$, the Jacobean of $\{f_{i_1}, f_{i_2}, ..., f_{i_l}\} : R^d \to R^l$ has rank l at every point a of the solution set $\{a \in R^d : f_{i_1}(a) = f_{i_2}(a) = ... = f_{i_l}(a) = 0\}$.*

For instance, if two zero-sets 'touch' at a point, so that the hyperplanes tangential to them at that point coincide, the functions do not have regular zero-set intersections.

Definition 3.4.3. *A set G of real valued functions defined on R^{d}. We say that G has solution set components bound B if for any $1 \leq k \leq d$ and any $\{f_1, ..., f_k\} \subset G$ that has regular zero set intersection, we have $CC(\cap_{i=1}^{k}\{a \in R^d : f_i(a) = 0\}) \leq B$.*

The intersection of any $k > d$ zero-sets of functions with regular zero-set intersections must be empty. We shall always be concerned with classes F of real-valued functions defined on $\mathrm{R}^{\mathrm{d}} \times X$, and with the solution set components bound for the class $G = \{a \to f(a,x) : f \in F, x \in X\}$. Furthermore, we say that F is closed under addition of constants if, for any $c \in R$, whenever $f \in F$, the function $(a,x) \to f(a,x) + c$ is also in F.

Theorem 3.4.1 ([14]). *Suppose that F is a class of real-valued functions defined on $\mathrm{R}^{\mathrm{d}} \times X$, and that H is a k-combination of $sgn(F)$. If F is closed under addition of constants, has solution set components bound B, and functions in F are C^{d} in their parameters, then $\Pi_H(m) \leq B\sum_{i=0}^{d}\binom{mk}{i} \leq B(emk/d)^d$, for all $m \geq d/k$.*

Proof. Taking zero sets Z_i in Lemma 7.9 in [14] to be those of the mk functions $a \to f_i(a, x_j)$ defined as parameter space $\mathrm{R}^{\mathrm{d}}, i = 1(1)k, j = 1(1)m$. Then $\Pi_H(m) \leq \max_{Z_i}\sum_{S\subset\{1,2,...,mk\}} CC(\cap_{i\in S} Z_i)$. This is less then equal to $B\sum_{j=0}^{d}\binom{mk}{j}$. Which follows from the fact that the intersection of more than d such zero sets is always empty. □

Example 3.4.1. *Suppose H is a class of functions computed by perceptron on R^{d}. Then the parameter space is $\mathrm{R}^{\mathrm{d}+1}$ and we can define F as the class of functions satisfying $f(a,x) = \sum_{i=1}^{d} x_i a_i + a_0 + c$ for some $c \in R$, where $a = (a_0, a_1, ..., a_d)$. In this case, F has solution set components bound $B = 1$.*

In the proof of the growth function bound for the perceptron, we first related the number of dichotomies of a set of input points x_i to the number of cells in the partition of the parameter space defined by the equations $w^T x_i - \theta = 0$.

For the proof of growth function bound for perceptron the author related the number of dichotomies of a set of input points x_i to the number of cells in the partition of the parameter domain defined by the equations $w^T x_i - \theta = 0$. The following lemma shows that we can do this more generally, for any class that is a k-combination of thresholded real-valued functions. In this case, we relate the growth function to the number of connected components of the complement of certain zero-sets of functions that have regular zero-set intersections.

Lemma 3.4.1 ([14]). *Given a set $\{f_1, f_2, .., f_k\}$ of C^{d} functions mapping from R^{d} to R, the set $S = \{\lambda \in \mathrm{R}^{\mathrm{k}} : \{f_1 - \lambda_1, .., f_k - \lambda_k\}$ does not have regular zero intersection$\}$ has measure 0.*

Let $F = \{f(., a) : a \in A\}$, A is a open subset of R^{m}, f is continuously differentiable. Let $g : A \times X^m \to \mathrm{R}^{\mathrm{m}}$ be defined by $g(a, x_1, ..., x_m) = (f(a, x_1), ..., f(a, x_m))^T$. For a fixed x, define $g_x(a) = g(a, x)$.

Theorem 3.4.2 ([22]). *Let A be an open subset of R^{m} and X be an open subset of R^{n} and $f : A \times X \to \mathrm{R}$ be a continuously differentiable function. Let $F := \{f(a, .) : a \in A\}$. If there exists a k dimensional manifold $M \subset A$, which has unique decision boundaries, then $VC(F_\theta) \geq k$.*

Lemma 3.4.2 ([14]). *Let F be a class of real-valued functions defined on $\mathrm{R}^{\mathrm{d}} \times X$ that is closed under addition of constants. Suppose that the functions in F are continuous in their parameters and let H be a k-combination of $sgn(F)$. Then for some functions $\{f_1, f_2, ..., f_k\}$ in F and some examples $x_1, .., x_m$ in X, the set $\{a \to f_j(a, x_i) : i = 1(1)m, j = 1(1)k\}$ has regular zero set intersections and the number of connected components of the set $\mathrm{R}^{\mathrm{d}} - \cup_{i=1}^{k} \cup_{j=1}^{m} \{a \in \mathrm{R}^{\mathrm{d}} : f_i(a, x_j) = 0\}$ is at least $\Pi_H(m)$.*

3.4.2 VC Bounds for Neural Networks

Here we discussed VC dimension of some deep neural networks using the above approach, we just have discussed. We first consider the classes of functions which can be expressed as a Boolean combination of thresholded real valued functions, each of which is polynomial in its parameters. We need a solution set component bound to apply Theorem 3.4.1. For this purpose we need the following theorem.

Theorem 3.4.3 ([14]). *Let F be a class of functions mapping from $\mathrm{R}^{\mathrm{d}} \times \mathrm{X}$ to R so that, for all $x \in X$ and $f \in F$, the function $a \to f(a, x)$ is a polynomial on R^{d} of degree no more than l. Suppose that H is a $k-$combination of $sgn(F)$. Then if $m \geq d/k$, $\Pi_H(m) \leqslant 2(\frac{2emlk}{d})^d$ and hence $VC(H) \leqslant 2d \log_2(12kl)$.*

The above mentioned theorem can be used to give bounds on the VC dimension of a function class in terms of the number of arithmetic operations required to compute the functions, as the following theorem demonstrates.

Theorem 3.4.4 ([14]). *Suppose h is a function from $\mathrm{R}^{\mathrm{d}} \times \mathrm{R}^{\mathrm{m}}$ to $\{0, 1\}$ and let $H = \{x \to h(a, x) : a \in \mathrm{R}^{\mathrm{d}}\}$ be the class determined by h. Suppose that h can be computed by an algorithm that takes as input the pairs $(a, x) \in \mathrm{R}^{\mathrm{d}} \times \mathrm{R}^{\mathrm{n}}$ and returns $h(a, x)$ after no more than t operations of the following types:*

- *The arithmetic operation* $+, -, *, /$ *on real numbers.*
- *Jumps condition on* $>, \geq, <, \leq, =, \neq$ *comparison of real nos.*
- *output* 0 *or* 1.

Then $VC(H) \leqslant 4d(t+2)$.

The following theorem shown that the bound of Theorem 3.4.4 can not be improved more than a constant factor.

Theorem 3.4.5 ([14]). *For all* $d \geq 1$, *there is a class* H *of functions, parametrized by* d *real numbers, that can be computed in time* $O(t)$, *and that has* $VC(H) \geq dt$.

As an application of Theorem 3.4.4 we may consider a class of feed forward linear threshold networks. Since the computing a output of linear threshold network takes time $O(w)$, so the following bound is worse than the bound $O(w \log(w))$. The theorem can be generalized a network with piecewise polynomial functions.

Theorem 3.4.6 ([14]). *Suppose* N *is a feed forward linear threshold network with a total of* w *weights, and let* H *be the class of functions computed by this network. Then* $VC(H) = O(w^2)$.

Theorem 3.4.7 ([14]). *Suppose* N *is a feed forward network with a total of* w *weights and* k *computation units, in which the output is a linear threshold unit and every other computation unit has a piece wise polynomial activation functions with* p *pieces and degree no more than* l. *Then if* H *is the class of functions computed by* N, $VC(H) = O(w(w + kl \log_2 p))$.

Proof. To compute an activation functions, we can determine the appropriate piece with $\log_2 p$ comparisons. Computing the value of the function takes an additional $O(l)$ steps. Hence total computation time is $O(w + kl \log_2 p)$. □

The author construct a network of linear threshold units and linear units. This construction is help full for the below theorem. The theorem shows that the bound $O(w^2)$ can not be improved more than by a constant factor if we allow a arbitary number of layers.

Theorem 3.4.8 ([14]). *Suppose* $s : \mathrm{R} \to \mathrm{R}$ *has the following properties:*

- $lim_{\alpha \to \infty} s(\alpha) = 1$ *and* $lim_{\alpha \to -\infty} s(\alpha) = 0$
- s *is differentiable at some point* $\alpha_o \in \mathrm{R}$ *with* $s'(\alpha_o) \neq 0$.

For any $L \geq 1$ *and* $w \geq 10L - 14$, *there exists a feed forward network with* L *layers and total* w *parameters, where every computation unit but the output unit has activation functions* s, *the output unit being a linear threshold unit, and for which the set* H *of functions computed by the network has* $VC(H) \geq (L/2)(w/2)$.

The below theorem holds for two layered neural network with sigmoid activation functions. But input of this network is discrete. The author defined the fan-in of a computation unit to be the number of input units or computation units that feed into it.

Theorem 3.4.9 ([14]). *Consider a two layer feed forward network with input domain* $X = \{-D, -D + 1, ..., D\}^n$ *for* $D \in N$ *and* k *first layer computation units, each with the standard sigmoid activation function (output L.T.). Let* w *be the total no of parameters in the network, and suppose that the far-in of each first layer unit is no more than* N. *Then the class* H *of functions computed by this network has* $VC(H) \leqslant 2w \log_2(60ND)$.

Proof. For a first layer unit, $x_1, ..., x_N$ be the input and $w_1, .., w_N$ are the corresponding weights and θ be the threshold. So the unit computes $f(x) = 1/(1 + exp(-\sum_{j=1}^{N} w_j x_j + \theta))$. It computes the *sgn* function of an affine combination of k of these rational functions or of inputs. □

Theorem 3.4.10 ([37]). *Consider a two layer feed forward linear threshold network that has w parameters and whose first layer units have far-in no more than N. If H is the set of functions computed by this network on binary inputs, then $VC(H) \leqslant 2w \log_2(60N)$.*

The following theorem provides a general VC dimension bound for standard sigmoid network. There is a considerable gap between this bound $O(kw^2)$ and the lower bound $\Omega(w^2)$, which is exhibited by a neural network with $k = \Theta(w)$ computation units.

Theorem 3.4.11 ([14]). *Let H be the set of functions computed by a feed forward network with w parameters and k computation units, in which each computation unit other than the output unit has the standard sigmoid activation function. Then $VC(H) \leqslant (wk)^2 + 11wk \log_2(18wk^2)$.*

The below theorem is a counter part of the Theorem 3.4.4. In this case, the author also allow the computation of the exponential function to be one of the basic operations.

Theorem 3.4.12 ([14]). *All the conditions same as Theorem* 3.4.4 *and to addition $\alpha \to exp(\alpha)$ on real numbers. Then $VC(H) \leqslant t^2 d(d + 19 \log_2(9d))$.*

This result immediately implies a bound on the VC dimension for feed forward standard sigmoid networks that is only a constant factor worse than the bound of Theorem 3.4.11. To proof Theorem 3.4.12 we need the solution set component bound for polynomial of certain exponential function. For this purpose we need the following theorem.

Theorem 3.4.13 ([14]). *Let $f_1, ..., f_q$ be fixed affine functions of $a_1, ..., a_d$ and let h be the class of polynomials in $a_1, ..., a_d, exp(f_1(a)), ..., exp(f_q(a))$ of degree no more than l. Then h has solution set components bound $B = 2^{q(q-1)/2}(l + 1)^{2d+q}(d + 1)^{d+2q}$.*

Lemma 3.4.3 ([14]). *Suppose G is the class of functions defined on R^{d} computed by a circuit satisfying the following conditions: the circuit contains q gates, the output gate computes a rational function of degree no more than $l \geq 1$, each non output gate computes the exponential function of a rational function of degree no more than l, and the denominator of each rational function is never zero. Then G has solution set components bound $2^{(qd)^2/2}(9qdl)^{5qd}$.*

3.5 Application of VC Dimension on Machine Learning

The performance of learning machine on test data is called generalization performance of a machine. For a given learning task, with finite set of training examples the best generalization will be achieved if the right balance will be stucked between the accuracy attained on that particular training set and the capacity (expressiveness) of the machine. VC Dimension comes from a similar concept in the information theory. The observation is if you have N objects and among those N objects you are looking for a specific one. How many bits of information do you need to find this object. Suppose you have N functions such that given input x, you have to find how many functions give you yes and how many give you no. How many training examples do you need to remove all those wrong functions. A machine with more capacity could give low

training error, but might over fit the data as a result of gives low performance on test data. A machine with less capacity not going to over-fit, but restricted in what it can model. How can we characteristics the capacity of learning machines? VC Dimension provides a quantitative way to measure the capacity of a learning machine.

3.5.1 PAC Model

Generate instances from unknown distribution p, $x^i \sim p(x), \forall i$. Oracle labels each instance with unknown function c, $y^i = c(x^i), \forall i$. Learning algorithm chooses hypothesis $h \in H$ with low training error $R\hat{(}h)$, $\hat{h} = argmin_h R\hat{(}h)$. Our goal is to choose an h with low generalization error $R(h)$. Define True Error (Expected Risk) is $R(h) = p_{x\sim p(x)}(c(x) \neq h(x))$, Train Error (Empirical Risk) is $R\hat{(}h) = p_{x\sim S}(c(x) \neq h(x))$, where S is a training set. And also Expected Risk Minimization (Lower True Error) is $h' = argmin_{h\in H} R(h)$ and Empirical Risk Minimization (Lower Training Error) is $\hat{h} = argmin_{h\in H} R\hat{(}h)$. The goal of the model is to learn a concept so that with a high degree of confidence the prediction error will be small. A learning machine or concept classes defined as a set of possible mapping $x \mapsto f(y, (x, a))$ where x is in input domain, y is the labels and a is the parameter. A particular choice of a gives a trained machine. PAC approach is that the error should not depends on the data distribution. The bound is the distribution free. The concept of VC Dimension is distribution free.

VC Dimension and Number of Parameters

The VC Dimension give the concreteness of the notion of the capacity of a given concept class.Intuitively one might expect that learning machines with more parameters would give high VC Dimension, while learning machines with less parameters would has low VC Dimension. Although this is true for most cases, some counter examples exists. VC Dimension is responsible for how many example need to learn and PAC learning responsible for how many mistakes before you converges.

3.5.2 VC Dimension and Learnability

It helps to answer some questions on learning theory like (a) Is a concept class learnable (b) Can a concept class learned efficiently (c) How many training samples do we need. The PAC criteria is that the learner produces a high accuracy learner with high probability. Algorithm consistent if for all $\epsilon, \delta > 0$, there exists N training examples such that for any distribution p, we have $p(|R(h) - R\hat{(}h)| \leq \epsilon) \geq 1 - \delta$. The sample Complexity is the minimum value of N for which this statement holds.

Definition 3.5.1. *The Static learning algorithm has the following properties: the number of samples it ask for is PAC Bound and It chooses its hypothesis based on the sample it gets.*

But Static learning algorithm is not adaptive. The below two Theorem related to Static learning algorithm.

Theorem 3.5.1 ([19]). *If a concept class C has ∞ VC Dimension, then C is not learnable by any Static learning algorithm.*

Theorem 3.5.2 ([19]). *The concept class $C(n)$ is not polynomially learnable, if the VC Dimension of $C(n)$ grows more than polynomial in n, where n is the dimension of domain space.*

Theorem 3.5.3 (Upper Bound on Sample Complexity (Blumer 1989)). *Let H and F be two function classes such that $F \subset H$ and let A an algorithm that derives a function $h \in H$ consistent with m training examples. Then there exists c_0 such that for all $f \in F$, for all D distribution, for all $\epsilon > 0$ and $\delta < 1$ if $m > \frac{c_0}{\epsilon}(VC(H).\ln\frac{1}{\epsilon}.\frac{1}{\delta})$, then with a probability $1-\delta$, $error_D(h) \leq \epsilon$, where $error_D(h)$ is the error of h according to the data distribution D.*

Theorem 3.5.4 (Lower Bound On Sample Complexity (Blumer 1989)). *To learn a concept class F whose VC Dimension is d, any PAC algorithm requires $m = O(\frac{1}{\epsilon}.(d + \frac{1}{\delta}))$ examples.*

Theorem 3.5.5 (Bound on Classification Error (Vapnik 1995)). *Let H be a hypothesis space having VC Dimension d. For any probability distribution D on $X \times \{0,1\}$, with probability $1-\delta$ over m random examples S, any hypothesis $h \in H$ that is consistent with S has error no more than $error(h) \leq \epsilon(m, H, \delta) = \frac{2}{m}(d + \ln\frac{2}{\delta}.\ln\frac{2em}{d})$, provided that $d \leq m$ and $m \geq 2/\epsilon$.*

3.5.3 VC Dimension and Generalization Performance

[19] A low complexity model will have a high bias and a low variance, while it has low expressive power leading to high bias, it is also very simple, so it has very predictable performance. Model with higher VC Dimension will require more train data to properly train. The Generalization performance concerns the error rate of a learning machine on test data.

Choose $0 \leq \eta \leq 1$. With probability $1-\eta$ the bound holds $R(\alpha) \leqslant R_{emp}(\alpha) + \sqrt{\frac{h(\log(2l/h)+1)-\log(\eta/4)}{l}}$. l is the number of training samples and h is the VC Dimension of the learning machine. $R(\alpha)$ is the expectation of test error. The quality of $R(\alpha)$ is called actual risk. The empirical risk $R_{emp}(\alpha)$ is the measure of mean error rate on training samples. $R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(x_i,\alpha)|$ and $R(\alpha) = \int(\frac{1}{2}|y - f(x,\alpha)|)dD(x,y)$, where $D(x,y)$ is the cumulative distribution that generates training and test set samples. The upper bound of actual risk known as VC Bound and second term of this bound known as VC confidence. This bound gives a principal method for choosing a learning machine for a given task and is the essential idea of the Structural Risk Minimization. Given a fixed family of learning machine, to choose from, to the extent that the bound is tight for at least one of the machines, one will not be able to do better than this.

3.5.4 Structural Risk Minimization

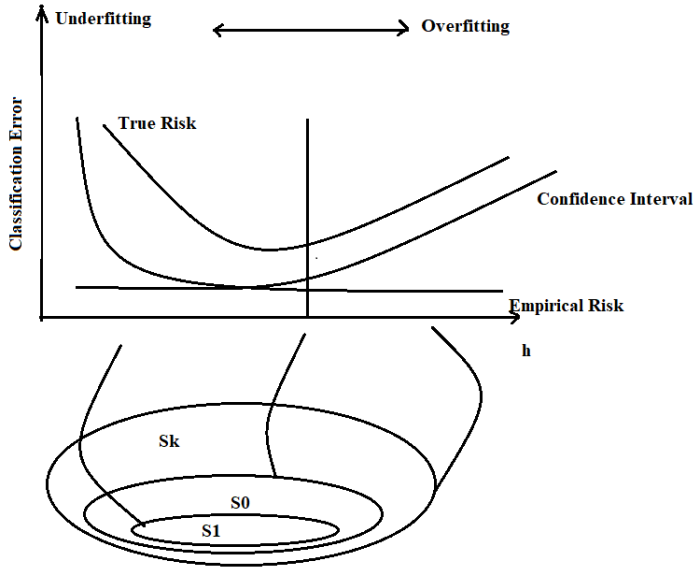


Figure 3.2: Comparison of Empirical Risk and True Risk

Each S_i have similar VC Dimension. VC Dimension is measured on the X axis as h. As complexity increase your transition from under fitting to over fitting, adding complexity is good up until a certain point. This approach suggest when we do choose a model. Only empirical error is not sufficient as for some machine may overfit training data. So consider VC Bound. VC Confidence depends on the chosen class of functions but actual risk and empirical risk only depends on the particular chosen function during training procedure. We would like to find the set of functions such that the risk bound for this set is minimized. To do that we divide the functions as some subset of functions such that functions for each class have same VC Dimension. Since all the functions within a subset have same VC Confidence, it is enough to compute only Empirical Risk of each machine. From each subset we choose the one which has minimum empirical risk. One then takes that trained machine in the series whose sum of empirical risk and VC Confidence is minimal.

3.5.5 Decision Tree

Decision tree are enough to express any Boolean valued function. We try to find a Decision tree which has smaller length and consistent on training samples. Finding the smallest Decision tree is a intractable problem. Decision tree pruning consists of methods Structural Risk Minimization, cross validation, $C4.5$. We need to compute VC Dimension h for a given tree. Roughly h is the number of internal nodes of a tree. The main problem is finding the best (Minimum Empirical Risk) decision tree for a given h. In a rigorous way we can do from all permutations of trees, but it is exponential time consuming procedure. So for overcome this difficulty we use a algorithm given in [19] using idea of dynamic programming.

Chapter 4

Random Vector Functional Link Network

This chapter gives an informal introduction to RVFLN and the class of problems we focus on in this dissertation.

4.1 Feedforward Neural Network (FNN)

The neurons in the adjacent layers are connected. But there is no interconnection of neurons within the same layer or across non-adjacent layers. In the input layer, each neuron i_m, $m \in \{1, 2, ..., M\}$ takes a feature of input vector and passes to the several hidden layers. Each neuron in $n'th$ hidden layer h_{n,k_n},$k_n \in \{1, 2, ..., K_N\}, n \in \{1, 2, .., N\}$ is formed by a nonlinear weighted sum of the outputs of the input layer or preceding hidden layer (except the last hidden layer).

$$h_{1,k_1} = f(\sum_{m=0}^{M} w_{m,k_1} i_m), \forall k_1 \in \{1, 2, ..., K_1\}.$$

$$h_{n,k_n} = f(\sum_{k_{n-1}=0}^{K_{n-1}} w_{k_{n-1},k_n} h_{n-1,k_{n-1}}), \forall k_n \in \{2, 3, ..., K_n\}.$$

$f(.)$ is a non linear activation function. $w_{0,k_n} = 1, k_n \in \{1, 2, ..., K_n\}$ denotes the input layer and hidden layer biases. M is number of input layer neurons, N is number of hidden layers, K_n is number of $n'th$ hidden layer neurons. w_{m,k_1} are weights between input and hidden layer neurons, w_{k_{n-1},k_n} are weights between the hidden layer neurons.

$$logsig(x) = 1/(1 + e^{-x})$$

$$tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$$

$$o_l = \sum_{k_N=0}^{K_N} w_{k_N,l} h_{N,k_N}, \forall l \in \{1, 2, ..., L\}.$$

L is number of output neurons, $w_{k_N,l}$ are weights between hidden and output layer neurons. To get optimal output value, the weights are determined by the BP learning. BP has a tendency to trapped in local minimum.

4.2 Single Hidden Layer Neural Network (SLFN)

SLFN has single hidden layer, adjacent layers connection, no interconnection of neurons with same layer or across non adjacent layers.

Method	ILB	HLB	in-out connection
M 1	p	p	p
M 2	p	a	p
M 3	a	p	p
M 4	a	a	p

Table 4.1: RVFL Network with Different Configurations

This table has taken from [33]. ILB: Input Layer Bias, HLB: Hidden Layer Bias, p: Present, a: Absent.

4.3 Random Weight SLFN (RWSLFN)

Schmidt reported SLFN with fixed random weights assigned to input to hidden layer. Hidden layer activation function is logsig. Training a SLFN is to minimize the squared output error by finding the optimal hidden layer weights $W_h = \{w_{m,k}, b_{i,k}\}$ and output layer weights $W_0 = \{w_{k,l}, b_{h,l}\}$.

$$min\ \epsilon^2 = \sum_{i=1}^{N}(y_i - \sum_{k=1}^{K} w_{k,l} f(\sum_{m=1}^{M} w_{m,k} i_m + b_{i,k}) + b_{h,l})^2.$$

For SLFN, the optimal hidden layer and output layer weights determined by BP. For RWSLFN, W_h are randomly sampled from a uniform distribution in $[-1, 1]$ and W_0 are optimized by a least square method.

4.4 Random Vector Functional Link Network (RVFLN)

The idea of this type of network has taken from [33]. It combines the advantage of random weights and functional link. It is an SLFN with direct connection from input layer to output layer. Enhancement nodes equivalent to hidden layer nodes. Use conjugate BP to tune the weights from input to output and enhancement to output layer. Apply least square method if matrix inversion is possible. Pao's - activation function in enhancement nodes is logsig, Chen's - hidden nodes use tanh, an additional activation function to output layer. Does not require iteratively updating the input to hidden layer weights, speed up the training process.

4.5 Equations

Equations are gien in the paper [33]. Related weight updating equations of RVFLN are..

E 1:

$$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m + b_{i,k})$$

$$o_l = \sum_{k=1}^{K} w_{k,l} h_k + b_{h,l} + \sum_{m=1}^{M} w_{m,l} i_m + b_{i,l}$$

E 2:

$$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m + b_{i,k})$$

$$o_l = \sum_{k=1}^{K} w_{k,l} h_k + \sum_{m=1}^{M} w_{m,l} i_m + b_{i,l}$$

E 3:

$$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m)$$

$$o_l = \sum_{k=1}^{K} w_{k,l} h_k + b_{h,l} + \sum_{m=1}^{M} w_{m,l} i_m$$

E 4:

$$h_k = f(\sum_{m=1}^{M} w_{m,k} i_m)$$

$$o_l = \sum_{k=1}^{K} w_{k,l} h_k + \sum_{m=1}^{M} w_{m,l} i_m$$

The below figure describes the structure of a RVFLN with one hidden layer. The figure has taken from [33].

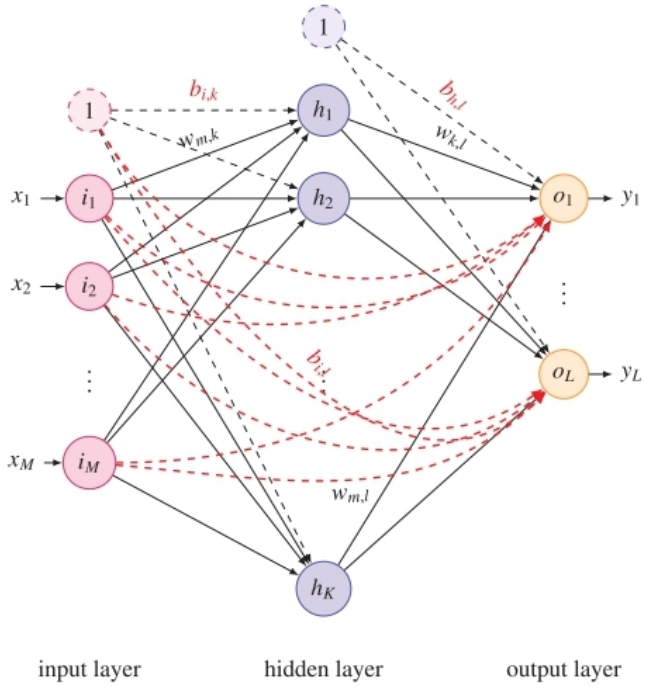


Figure 4.1: Random Vector Functional Link Network

Chapter 5

Related Work and Our Contribution

5.1 VC Dimension for Neural Network with Continuous Activation Functions

This section shows that neural network with continuous activation functions have VC dimension at least $O(w^m)$, where $O(w^{m-1})$ is the total parameters (weights and biases) of the network and also input domain is R^{m} for $m \geq 2$. For a satisfactory result on test data of a classifier we need to learn the classifier properly and accurately. For this purpose we need sufficient training samples. In computation Learning Theory branch, PAC formulation is a concept where we can determine number of examples needed to learn a classifier such that it will predict all future data correctly. If F is a hypothesis space or collection of binary valued functions, then we fit the training examples to each $f \in F$. We will consider the training data and test data from same domain X and follow same probability distribution. After the Vapnik's contribution in statistics, people know that a certain quantity known as VC Dimension, which is related to sample size. And also it is needed for learnability in the PAC sense. Roughly speaking, generally VC is proportional to the sample size needed for learn the machine reliably. Basically we focus on calculating VC of F. For the pure hard threshold class $VC(F) = O(w \log(w))$ by Cover, by Baum and Hausseler. For sigmoidal class the VC dimension is $O(w^4)$ by Karpinski and Macintyre and also for piece wise polynomial feedforward neural network VC dimension is $O(w^2)$. Also Mass showed that in [18] that there is also a lower bound of the form $\Omega(w \log w)$. Basically now days back propagation method rely upon continuous activation function, that is neuron with graded responses. Basically use analog activation give advantage of passing the rich information among layers. It needs higher memory capacity means to learn f we need more train data. This section showed that there are conceivable neural network architecture with exactly high VC Dimensions. Thus the study of $VC(F)$ of analog neural network is an interesting and relevant issues.

Jerrum and Goldberg showed that upper bound of VC is $O(w^2)$ for piece wise polynomial activation function. Now the question is Is there any neural network architecture which can achieve lower bound w^2 for such networks and more generally for arbitary continuous activation nets. For pure threshold nets VC proportional to $w \log w$ and pure linear nets VC proportional to w. Then there are architecture with

arbitary large number of weights w and VC proportional to w^2. First we are showing that the network with linear and Heaviside activation functions have this power. The continuous activation functions σ have the property in $+\infty, -\infty$ attain two different values and for at least one point it has derivative non zero. Now we obtained our expected results on continuous activation function, approximating Heaviside gates by σ nets with large weights and approximating linear gates by σ nets with small weights. However there is still a gap between $\Omega(w^2)$ lower bound and the $O(w^4)$ upper bound in [2]. The original paper [18] showed that real number program with running time T have VC Dimension $\Omega(T^2)$ for input space R^2. We mainly consider the case which generalize the input domain from R^2 to R^m, $m \geq 2$. And also accordingly our results will changed. When the input domain dimension is increased the VC dimension increased accordingly total number of weight increase. But out target will be for any dimensional input domain there exists a neural net architecture which has VC at least square of total number of weights of the network.

5.1.1 For Linear and Threshold Gates

All the notations has taken from [2]. We define an architecture or network $\mathbf{A}$ is a connected directed acyclic graph. In the network, a subset of nodes has a activation function or functions. One is identity or linear gate $id(x) = x$, and another one is threshold or Heaviside gate $H(x) = 1, x \geq 0$ and $H(x) = 0, x < 0$. F is collection of functions computed by the network A. For a given weight $w \in \mathrm{R}^n$, there exists a function $F_w : \mathrm{R}^m \to \mathrm{R}^p$ defined by $F_w(x) = F(x, w)$, sometimes we say that this function obtained from A's calculation. We say a subset $S \subset \mathrm{R}^m$ is shattered by A if for any arbitary Boolean function $\beta : S \to \{0, 1\}$ there exists some weight $w \in \mathrm{R}^n$ so that $F_w(x) = \beta(x)$ for all $x \in S$. If the output is real number then we use a thresholding in output node with respect to some real number, which gives us a Boolean output. If A is the net $w_0 + w_1 H(2x - 1)$, it has one linear gate and one Heaviside gate, input is x, number of weights if 4, which are $2, -1, w_0, w_1$. The phrase "for each $n \geq 1$ there is an architecture $\mathbf{A}$ with $O(n^{m-1})$ weights and gates in $S = \{id, Heaviside\}$" to assert the existence of a sequence of architectures $\mathbf{A}_n$ so that S is a set of gates for each $\mathbf{A}_n$ and so that the number of weights of $\mathbf{A}_n$ is $O(n^{m-1})$. $\mathbf{A}$ shatters a set of size $\theta(n)$ we really mean that there is a sequence of sets A_n so that $\mathbf{A}_n$ shatters A_n and the cardinality of each A_n is $\theta(n)$.

Theorem 5.1.1. *For every $n \geq 1$, there is a network architecture A with inputs in $\mathrm{R}^m (m \geq 2)$ and $O(n)$ weights that can shatters a set of size n^2. This architecture is made of linear and threshold gates.*

Proof. The shattered set S constructed in a sequential manner, described below and the construction follows the idea of [2]. If input in R^m and the network has n weights $W_1, ..., W_n$ where each W_i belongs to $T = \{0.w_1 w_2 ... w_n : w_i \in \{0, 1\}\}$, then the cardinality of the shattered set S will be n^2. And the elements are $\{(x_1, ..., x_m) : 1 \leq x_1 \leq n \; and \; for \; each \; x_1 \; 1 \leq x_m \leq n, \forall i \neq 1, m \; x_i = x_m\}$. Consider the example for $m = 4, n = 4$. Then the shattered set will have $n^2 = 16$ elements and the elements are $\{\{1111, 1222, 1333, 1444\}, \{2111, 2222, 2333, 2444\}, \{3111, 3222, 3333, 3444\}, \{4111, 4222, 4333, 4444\}\}$. Now for a given choice of $W = (W_1, ..., W_n)$, A will compute the Boolean function $g_W : S \to \{0, 1\}$ defined as $g_W(x_1, ..x_m) = x'_m th \; bit \; of \; W_{x_1}$. We have to show for any Boolean function g on S, there exists a unique W such that $g = g_W$.

A consists of three sub networks g^1, g^2, g^3. Now by our definition each of these three hold $O(n)$ weights. ${g^1}_W(i)$ will give output W_i, for all $1 \leq i \leq n$ and $W = (W_1, ..., W_n)$. $g^2(W_j)$ will give output $(w_1, w_2, ..., w_n)$, where $W_j = w_1 w_2 ..., w_n$ be a binary representation and $w_i \in \{0, 1\}$, $1 \leq j \leq n$. Output of $g^3(k, W_j)$ will be w_k, where $W_j = w_1 ..... w_n$ and $k \in [n]$, $j \in [n]$.

The obvious one architecture which computes the function:

$$g^1{}_W(i) = W_1 + \sum_{u=2}^{n}(W_u - W_{u-1})H(i - u + 0.5)$$

sending each $i \in [n]$ to W_i. It has one linear gate, $n-1$ threshold gates, $3(n-1)+1$ weights.

g^2 is a multi output net. Basically g^2 constructed using a sequence of $N^2{}_i$ nets, $1 \leq i \leq n$. $N^2{}_i$ takes as input W_j and produce output $(w_1, ..., w_i, 0.w_{i+1}....w_n)$ for $1 \leq j \leq n$. Since $0.w_{i+2}....w_n = 10.0.w_{i+1}....w_n - w_{i+1}$ and $w_{i+1} = H(0.w_{i+1}....w_n - 0.5)$ holds, from $N^2{}_i$ to $N^2{}_{i+1}$ we need extra one threshold gate, one linear gate, four weights. As of our knowledge, we can say $N_n = g^2$, and to compute g^2 total n linear gates, $4n$ weights, n threshold gates required.

And the last one is:

$$g^3(k, W_j) = w_1 + \sum_{u=2}^{n} w_u H(k - u + 0.5) - \sum_{u=2}^{n} w_{u-1} H(k - u + 0.5), 1 \leq j \leq n, k \in [n]$$

As multiplication of inputs are not allowed, so uv can be replaced by $H(u + v - 1.5)$. Then in our network ew can replace $w_u H(k - u + 0.5)$ by $H(w_u + H(k - u + 0.5) - 1.5)$. Thus g^3 has total $4(n-1)$ threshold gates, one linear gate, $12(n-1) + n$ weights.

Finally our original network is $g_W(x_1, ..., x_m) = g^3(x_m, g^2(g^1{}_W(x_1)))$. It can be constructed using $n+2$ linear gates, $(n-1) + 4(n-1) + n = 6n - 5$ threshold gates and $(3n-2) + 4n + (12n - 11) = 19n - 13$ weights. □

Theorem 5.1.2. *For every $n \geq 1$, there is a network architecture* **A** *with inputs in* $\mathrm{R^m}(m \geq 2)$ *and* $O(n^{m-1})$ *weights that can shatter a set of size n^m. This architecture is made only of linear and threshold gates.*

Proof. $W_1, .., W_n$ are n parameters of our architecture where each W_i's is a element of $T = \{0.w_1 w_2 .....w_{n^{m-1}} : w_i \in \{0, 1\}\}$. We have to show that $S = [n]^m = \{1, 2, ..., n\}^m$ will be the shattered set.

Suppose we have a predefined weight vector $W = (W_1, ..., W_n)$. Now for this vector the network **A** will compute the Boolean function $f_W : S \to \{0, 1\}$ and defined as follows: $f_W(x_1, x_2, ..., x_m)$ is equal to the qth bit of W_{x_1}. Our target is for any Boolean function$f$ on S, there must exists a unique W such that $f = f_W$. Now we take $q = [\sum_{i=0}^{m-2} n^i(n - x_{m-i})] + 1$.

We consider a architecture which computes the function $f_W{}^1(x_2, x_3, ..., x_m)$, which gives the output $[\sum_{i=0}^{m-2} n^i(n - x_{m-i})] + 1$. The inputs are $x_2, ..., x_m$, nodes in hidden layer are $u_1, ..., u_m$, and single output node. So net input of the node u_j is $net_j = \sum_{i=2}^{m} w_{ij} x_i, 2 \leq j \leq m$. The output of the node u_j is $out_j = id(net_j + n)$. w_{ij} implies edge from x_i to u_j. And also $w_{ij} = -1 \; if \; i = j, otherwise \; 0$. Bias of hidden layer is n. So net input of the output node is $net_{output} = \sum_{k=2}^{m} n^{m-k} u_k$ and output of output node is $id(net_{output} + 1)$, bias in output layer is $+1$ and weights are $n^{m-2}, ..., n^0$ from hidden layer to output layer. So the network is

$$f^1{}_W = id(\sum_{j=2}^{m} id(\sum_{i=2}^{m} x_i w_{ij} + n)n^j + 1)$$

where $w_{ij} = -1$ if $i = j$ for $i, j \in \{2, .., m\}$. So total linear gates is m, Heaviside gate is 0, total weights are $m^2 - m + 2$.

According to our condition $m^2 - m + 2 \leq k.n^{m-1}$, where $k < n$ is a positive constant. But this inequality always holds, because only possible case for contradiction is $n <<< m$. But also in this case n^{m-1} increases

exponentially with respect to $m^2 - m + 2$, which is basically a polynomial of m. So we conclude from here that ${f^1}_W$ has weights $O(n^{m-1})$.

Now we define second architecture

$$ {f_W}^2(x_1) = W_1 + \sum_{z=2}^{n} (W_z - W_{z-1}) H(x_1 - z + 1/2) $$

which computes for each point $x_1 \in [n]$ to W_{x_1}. This network has $n-1$ threshold gate, 1 linear gate, $3(n-1)+1$ weights.

We define a architecture which maps $W_i, 1 \leq i \leq n$ to $w_1, w_2, ..., w_{n^{m-1}}$. So basically it is a multi output net. Then the network would be $f^3(w) = (w_1, w_2, ..., w_{n^{m-1}})$.

Assume by induction that we have a net ${N^3}_i$ that maps w to $(w_1, .., w_i, 0.w_{i+1}...w_{n^{m-1}})$. Since $w_{i+1} = H(10.(0.w_{i+1}...w_{n^{m-1}}) - 1/2)$ and $0.w_{i+2}...w_{n^{m-1}} = 10.0.w_{i+1}...w_{n^{m-1}} - w_{i+1}$, ${N^3}_{i+1}$ can be obtained by adding one threshold gate and one linear gate to ${N^3}_i$, as well as 4 weights. It follows that $f^3 = {N^3}_{n^{m-1}}$ has n^{m-1} threshold gates, n^{m-1} linear gates, and $4n^{m-1}$ weights.

Finally we define a net N^4 which takes as input $q \in [n^{m-1}]$ and $w = (w_1, w_2, ..., w_{n^{m-1}}) \in \{0,1\}^{n^{m-1}}$, and outputs w_q. The network is as follow

$$ f^4(q, w) = w_1 + \sum_{z=2}^{n^{m-1}} w_z H(q - z + 1/2) - \sum_{z=2}^{n^{m-1}} w_{z-1} H(q - z + 1/2). $$

As multiplication between w_i and Heaviside function are not allowed, so instead of uv we write $H(u+v-1.5)$, as v, u are binary valued. Therefore N^4 has 1 linear gate, $4(n^{m-1}-1)$ threshold gate, $12(n^{m-1}-1)+n$ weights.

So

$$ f_W(x_1, .., x_m) = f^4(f^1(x_2, .., x_m), f^3({f^2}_W(x_1))). $$

This implies that the net has total $m+1+n^{m-1}+1$ linear gates, $(0+n-1+n^{m-1}+4(n^{m-1}-1))$ threshold gates, $((m^2-m+2)+3(n-1)+1+4n^{m-1}+12(n^{m-1}-1)+n)$ weights.

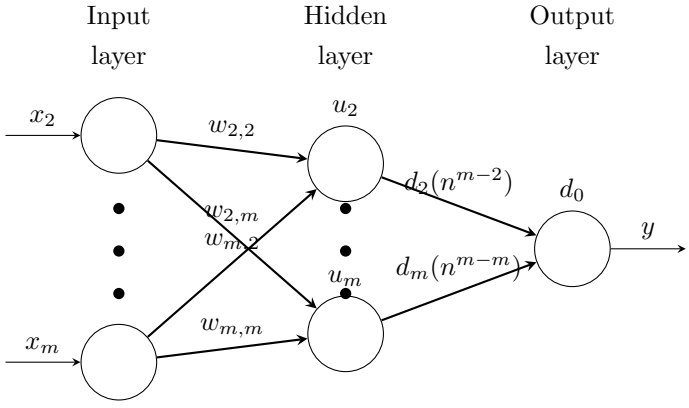


Figure 5.1: The network f^1, where input in R^{m} and shattered set is $[n]^m$.

This completes the proof. □

5.1.2 For Linear, Threshold, Multiplication, Division Gates

Lemma 5.1.1 ([2]). *For all $n \geq 1$, there exists an architecture A_1 with inputs $(x, W_1, W_2, ..., W_{n^{m-1}})$ in $\mathrm{R}^{\mathrm{n}^{\mathrm{m}-1}+1}$ and $O(n^{m-1})$ weights such that the following property holds: for every $\epsilon > 0$, there exists a choice of the weights of A_1 such that the function ${f^1}_\epsilon$ implemented by the network satisfies $\lim_{\epsilon \to 0} {f^1}_\epsilon(i, W_1, ..., W_{n^{m-1}}) = W_i$ for all $i = 1(1)n^{m-1}$.*

Proof. Let us consider the map ${f^1}_\epsilon(x, W_1, W_2, ..., W_{n^{m-1}}) = \Pi_{i=1}^{n^{m-1}}(x - \epsilon - i) \sum_{i=1}^{n^{m-1}} \frac{a_i * W_i}{x - \epsilon - i}$, where $a_i = \frac{1}{\Pi_{j \neq j}(i-j)}$ and $\epsilon \neq 0$. For this implementation we have used one multiplication gate with $n^{m-1} + 1$ inputs, one linear gate with n^{m-1} inputs, n^{m-1} division gates, n^{m-1} linear gate with one input each (compute the value of $x - \epsilon - i$). Total number of weights used here $n^{m-1} + 1 + n^{m-1} + 2n^{m-1} + 2n^{m-1} = 6n^{m-1} + 1$. Now ${f^1}_\epsilon(x, W_1, ..., W_{n^{m-1}}) = \sum_{i=1}^{n^{m-1}} W_i . \frac{\Pi_{j \neq i}(x - \epsilon - j)}{\Pi_{j \neq i}(i-j)}$. Hence $\lim_{\epsilon \to 0} {f^1}_\epsilon(x, W_1, ..., W_{n^{m-1}}) = W_x$. □

Lemma 5.1.2 ([2]). *There exists an architecture of linear and multiplication gates with inputs in R, n output units and $O(n)$ weights such that the following property holds for every $\epsilon \in \{0,1\}^n$, there exists an input $w \in [0,1]$ such that the output of the network $f^2(w) = (f^2(w)_1, ..., f^2(w)_n)$ of the network satisfies $f^2(w)_i \in [0, 1/2[, if\ \epsilon_i = 0$ and $f^2(w)_i \in]1/2, 1], if\ \epsilon_i = 1$.*

Proof. Consider the function $\phi : [0,1] \to [0,1]$ such that $\phi(x) = 4x(1-x)$. We claim that for all $\epsilon \in \{0,1\}^n$, there exists $w \in [0,1]$ such that $\phi^{i-1}(w) \in [0, 1/2[, \epsilon_i = 0$ and $\phi^{i-1}(w) \in]1/2, 1], \epsilon_i = 1$. This result follows from the claim, using the iterates $\phi^{i-1}, i = 1(1)n$ as the co ordinates of f^2, since the logistic map can be implemented by a sub-network of linear and multiplication gates.

Note that each element of $[0,1]$ has two distinct preimages by ϕ, except 1, and that $\phi(1/2) = 1, \phi(1) = 0, \phi(0) = 0$. If $\epsilon_n = 0$, choose an element $w_n \in]0, 1/2[$ otherwise choose $w_n \in]1/2, 1[$. We construct a sequence $w_1, w_2, ..., W_n$ by "going backward in time" as follows, w_i is defined to be the preimage of w_{i+1} which is in $]0, 1/2[$ if $\epsilon_i = 0$ and the preimage which is in $[1/2, 1[$ otherwise. By construction one can take $w = w_1$. □

Theorem 5.1.3. *For every $n \geq 1$, there is a network architecture with inputs in $\mathrm{R}^{\mathrm{m}}, m \geq 2$ and $O(n^{m-1})$ weights that can shatter a set of size n^m. This architecture is made only of linear, multiplication and division gates.*

Proof. First consider $[n]^m$ be our shattered set. And also assume f be an arbitary Boolean function on this shattered set. Let the input sequence of the net f^2 of Lemma 5.1.2 be $W = (W_1, ..., W_{N^{m-1}})$ and it satisfies $H({f^2(W_k)}_{x_1} - \frac{1}{2}) = f(x_1, x_2, ..., x_m) = f_1(x_1, k)$ for $x_i \in [n], i = 1(1)m$ and $k = [\sum_{i=2}^{m} n^{m-i}(x_i - 1)] + 1$. Now consider the map $N_\epsilon : (x_1, ..., x_m) \to {f^1}_\epsilon(x_1, f^2({f^1}_\epsilon(k, W))_1, ..., f^2({f^1}_\epsilon(k, W))_n)$. By Lemma 5.1.1 $\lim_{\epsilon \to 0} {f^1}_\epsilon(j, W) = W_j$. By continuity of f^2, when ϵ is small enough $f^2({f^1}_\epsilon(k, W))_i < 1/2$ if $f_1(x_1, k) = 0$ and $f^2({f^1}_\epsilon(k, W))_i > 1/2$ if $f_1(x_1, k) = 1$, for all $i = 1(1)n$. Hence it follows from Lemma 5.1.1 that when ϵ is small enough, $N_\epsilon(x_1, ..., x_m) < 1/2$ if $f_1(x_1, k) = 0$ and $N_\epsilon(x_1, ..., x_m) > 1/2$ if $f_1(x_1, k) = 1$. In a conclusion the Boolean function f thus be computed by comparing the output of N_ϵ to $1/2$. □

5.1.3 Conclusion

VC dimension for feed forward neural network with linear and threshold gates is square of total number of parameters. Thus we can not say that VC dimension upper bound of this network is $w \log w$. As already we have proved that there exists a network which has VC upper bound $O(w^2)$. Next we have showed that the VC dimension also depends upon the input dimension of the net. If input dimension increases, then size of shatter set will be increase along with total number of parameters increases. The size of shatter

set changes proportionally with number of parameters. As a part of the conclusion we can say that if we wants to increase the shattered set size for a neural network, then we have to increase also total number of programmable parameters.

5.2 VC Dimension for Piece wise Polynomial Network

Theorem 5.2.1. *If* $2^m \leq 2^t(mr/w)^w$ *where* $r \geq 16$ *and* $m \geq w \geq t \geq 0$. *Then* $m \leq t + w\log_2(\log_2 r.\log_2(\log_2 r))$.

Proof. We would like to show that $2^x > 2^t(xr/w)^w$ for all $x > t + w\log_2(\log_2 r.\log_2(\log_2 r))$. Let $f(x) = x - t - w\log_2(xr/w)$. To show that $f(x) > 0$ for all $x > m := t + w\log_2(\log_2 r.\log_2(\log_2 r))$. We need only to show that $f(m) \geq 0$ and $f'(x) > 0$ for all $x > m$. First $f(m) \geq 0$ iff

$$\begin{aligned}
w\log_2(\log_2 r.\log_2(\log_2 r)) - w\log_2(mr/w) &\geq 0 \\
iff\ \log_2(r).\log_2(\log_2 r) - mr/w &\geq 0 \\
iff\ \log_2(r).\log_2(\log_2 r) - \frac{t + w\log_2(\log_2 r.\log_2(\log_2 r))}{w} &\geq 0 \\
iff\ \log_2(r).\log_2(\log_2 r) - \frac{t}{w} - \log_2(\log_2 r.\log_2(\log_2 r)) &\geq 0 \\
iff\ \log_2(r + \log_2 r) - \frac{t}{w} - \log_2(\log_2(r + \log_2 r)) &\geq 0 \\
iff\ \log_2(\frac{r + \log_2 r}{\log_2(r + \log_2 r)}) - \frac{t}{w} &\geq 0 \\
iff\ \frac{r + \log_2 r}{\log_2(r + \log_2 r)} &\geq 2^{\frac{t}{w}}
\end{aligned}$$

Now to show $g(x) = \frac{x}{\log_2 x} \geq 2$ for $x \geq 20$. It is enough to show $g'(x) \geq 0$ for all $x \geq 20$.
Only remaining part is $f'(x) > 0$ for all $x > m$.

$$\begin{aligned}
f(x) &= x - t - w\log_2(xr/w) \\
f(x) &= x - t - w\log_2(xr) - w\log_2 w \\
f'(x) &= 1 - \frac{1}{x\log_e 2} \\
f'(x) &= 1 - \frac{10}{x.6.93} \\
f'(x) > 0 &\implies x > 1.44
\end{aligned}$$

So in addition an extra condition is $t + 3w > 1.44$. This completes the proof. □

Now we apply this result to our VC upper bound theorem in [16]. The proof of the rest part given in

this paper, so we only focus on a part where we can apply our result.

$$\begin{aligned}\Pi_{sgn(F)}(m) &\leq \Pi_{i=1}^{L} 2(\frac{2empk_i(1+(i-1)d^{i-1})}{W_i})^{W_i} \\ &\leq 2^L(\frac{2emp\sum k_i((i-1)d^{i-1}+1)}{\sum W_i})^{\sum W_i} \\ &= 2^L(\frac{2empR}{\sum W_i})^{\sum W_i} \\ &\leq (\frac{4emp(1+(L-1)d^{L-1})\sum k_i}{\sum W_i})^{\sum W_i} \\ &\leq (4emp(1+(L-1)d^{L-1}))^{\sum W_i}\end{aligned}$$

From the third line of this proof and definition of VC dimension, $2^{VC(F)} = \Pi_{sgn(F)}(VCdim(F)) \leq 2^t(\frac{2epRVCdim(F)}{\sum W_i})^{\sum W_i}$. Then the above theorem gives $VCdim(F) \leq L + (\sum W_i)\log_2(\log_2(2epR).\log_2(\log_2(2epR)))$, where $2epR = r, VCdim(F) = m, L = t, \sum W_i = w$.

5.3 VC Dimension of S shape functions

5.3.1 Neural Network with Sigmoid Activation Functions

Theorem 5.3.1. *Consider a neural network of W parameters, L layers and K computation units, except output unit all the units contain sigmoid activation function. The input of the network is real numbers and output is one unit with binary output. So for $L \leq W$ and $k \leq W$, the class of functions computed by this network is F. Then $VC(sgn(F)) \leq 2WL\log(2eWLpK) + 2WL^2\log(l+1) + 2L$, where l is max degree of the polynomial functions and p is the total breakpoints of that function. Also if l, p is fixed and $L, K = O(W)$, then $VC(sgn(F)) = O(WL\log W + WL^2)$.*

Chebyshev Approximation

Let $f(x)$ be a real valued function, we want to approximate to it such that $f(x) \approx \sum_{i=0}^{\infty} c_i T_i(x)$, where c_i's are the coefficients and T_i's are the normalized Bernstein basis functions. We will get the basis functions from the recursion $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$ where base case is $T_0 = 1, T_1 = x, n \geq 1$. First few polynomials are $T_2(x) = 2x^2-1$, $T_3(x) = 4x^3-3x$, $T_4(x) = 8x^4-8x^2+1$. Basically the Bernstein basis functions of degree n defined on $(0,1)$ is $b_{k,n}(t) = \binom{n}{k}t^k(1-t)^{n-k}$. But we need on interval $(-1,1)$, we give a transformation $s = 2t-1$ and the changed functions are $b_{k,n}(s) = \binom{n}{k}(1+s)^k(1-s)^{n-k} * 2^n$. Now the integral of normalized basis function becomes $C_{p,q}(s) = \frac{1+s}{2}^{q+1}\sum_{i=0}^{p}\binom{i+q}{i}\frac{1-s}{2}^i$. The expansion of Chebyshev polynomials follows $C_{p,q}(s) = \sum_{i=0}^{N} a(i)T_i(s)$, where $N = p+q+1$. The below figure gives a idea between original sigmoid and scaled sigmoid curve. The equation of sigmoid funtion is $f(x) = \frac{1}{1+e^{-x}}$ and equation of scaled sigmoid is $\sigma(x) = \frac{1}{1+e^{-8x}}$.

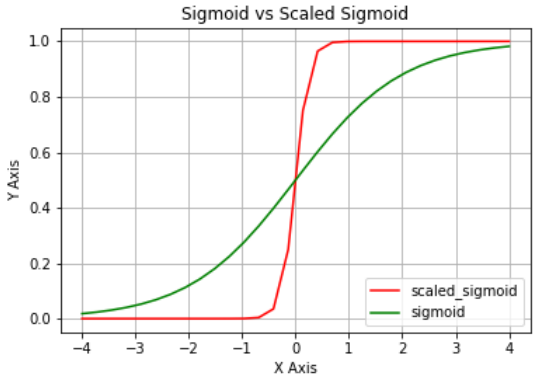


Figure 5.2: Comparison of sigmoid and scaled sigmoid functions

For scaled sigmoid function $\sigma(s) \approx C_{p,q}(s) = \frac{1+s}{2}^{q+1} \sum_{i=0}^{p} \binom{i+q}{i} \frac{1-s}{2}^{i} = \sum_{i=0}^{N} a(i) T_i(s)$ on interval $(-1, 1)$ and also choose $p = 11, q = 11$. So our approximation become $\sigma(x) = 0, x < -1$ and $C_{p,q}(x), -1 \leq x \leq 1$ and $1, x > 1$. The following figure shows a comparison between approximation polynomial and our original function.

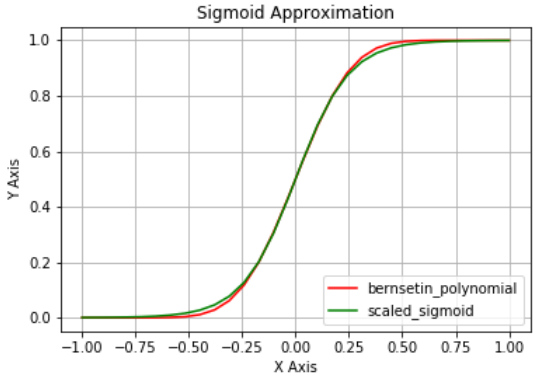


Figure 5.3: Sigmoidal Approximating Curve

VC Dimension Calculation of This Network

This activation function has maximum degree $l = 23$ and number of break points is $p = 3$. Choose m arbitary data points $x_1, x_2, ..., x_m$ from input domain and the parameter space is $S = \mathrm{R}^{\mathrm{W}}$. We have to bound $K := |\{(sgn(f(x_1, s)), ..., sgn(f(x_m, s))) : s \in S\}|$. Divide the parameter space such a way that in each part every polynomials have a fixed degree and no more than $(d+1)^{L-1}$. Partition S into $S_1, ..., S_t$ such that within each region $f(x_1, .), ..., f(x_m, .)$ are all polynomials of degree no more than $(d+1)^{L-1}$. Then $K \leq \sum_{i=1}^{t} |\{(sgn(f(x_1, s)), ..., sgn(f(x_m, s))) : s \in S\}|$. Then each term of the summation less than equal to $2(2em(d+1)^{L-1}/W)^{W}$. S_1 is determined by only parameters of first hidden layer. $W_1, ..., W_L$ be the parameters used in computing the unit outputs upto the layer $1, 2, ..., L$ respectively, and also $k_1, ..., k_L$ are the number of units upto the layer $1, 2, ..., L$ respectively. Choose S_1 such that $|S_1|$ is no more than the number of sign assignments possible with mk_1p affine functions with W_1 variables. Rest of the construction technique follows proof of [1]. Now applying our constraints $p = 3, l = 23$, which gives $VC(sgn(F)) \leq$

$2W\log(6eWLK) + 10WL^2 + 2L.$

Lagrange Approximation

Let $(x_1, f_1), (x_2, f_2), ..., (x_n, f_n)$ are the n choosen points and its corresponding functional values. Find polynomial $p(x)$ such that $p(x_1) = f_1, p(x_2) = f_2, ...$ Lagrange polynomials $L_1, L_2, ...$ have the following property:

$$L_1(x) = 1 \; at \; x = x_1 \; and \; L_1(x) = 0 \; for \; x = x_2, x_3, ..., x_n.$$

$$L_2(x) = 1 \; at \; x = x_2 \; and \; L_2(x) = 0 \; for \; x = x_1, x_3, ..., x_n \; and \; so \; on.$$

So our polynomial becomes $f(x) \approx p(x) = f_1L_1(x) + f_2L_2(x) + ...$ Where

$$L_1(x) = \frac{(x - x_2)(x - x_3)(x - x_4)...}{(x_1 - x_2)(x_1 - x_3)(x_1 - x_4)...}$$
$$L_2(x) = \frac{(x - x_1)(x - x_3)(x - x_4)...}{(x_2 - x_1)(x_2 - x_3)(x_2 - x_4)...}$$

Now Apply For Sigmoidal Network

For sigmoid $f(x)$ our approximate function will be

$$f(x) \approx p(x) = f_1L_1(x) + ...f_6L_6(x), where$$
$$L_1(x) = \frac{(x - x_2)(x - x_3)...(x - x_6)}{(x_1 - x_2)(x_1 - x_3)....(x_1 - x_6)}$$

and so on. So we basically approximate it by a 5 degree polynomial on $[-4, 4]$, and for less than -4 it will be 0 and for greater than 4 it will be 1. Our pre assumed 6 points are $x_1 = 1.050909826341572, x_2 = -3.2748296386052527, x_3 = 2.623648850589679, x_4 = -3.2990950890760633, x_5 = 3.522774111950934, x_6 = -2.1661034488309143$. And our comparison function picture will be

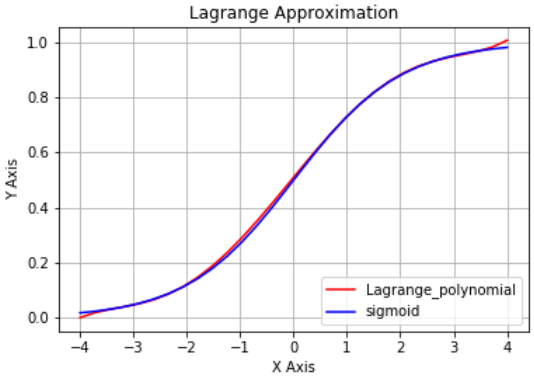


Figure 5.4: Lagrange Approximating Curve

Now apply Theorem 5.3.1 where our constraints would be $l = 5, p = 3$.

One More Approximation

Let $f(x)$ be our sigmoid function and we approximated it by $\sigma(x)$ where

$$\sigma(x) = 1 - \frac{1}{2x+2}, x \geq 0$$
$$= \frac{1}{2-2x}, x < 0$$

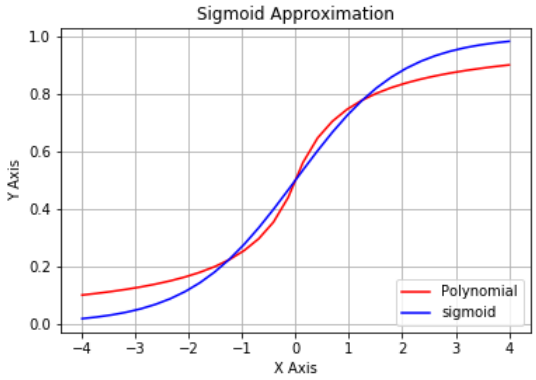


Figure 5.5: Another Approximation of Sigmoid

First check if Warren result for sign assignment of a polynomial valid for rational function, we will apply Theorem 5.3.1. Or our target is we approximate it by some polynomial approximation Theorem, next to apply Theorem 5.3.1.

5.3.2 Neural Network with tanh Function

The equation of tanh function is $f(x) = \frac{e^{2x}-1}{e^{2x}+1}$. If we compare tanh with sigmoid, this is a scaled version of one another. The below figure gives a idea of these two functions:

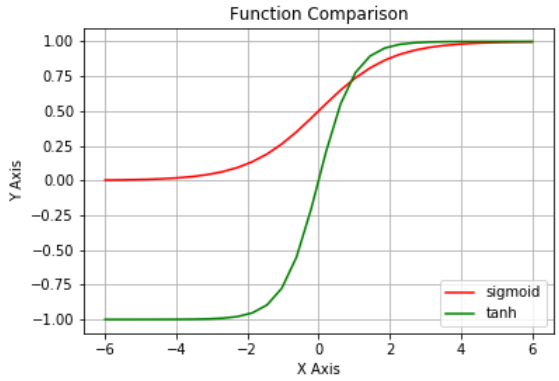


Figure 5.6: Comparison of sigmoid and tanh functions

Using Continued Fraction

The rational function that comes from truncation of the continued fraction for the hyperbolic tangent:

$$tanh(x) = \frac{x}{1 + \frac{x^2}{3 + \frac{x^2}{5 + \ddots}}}.$$

Here for instance are plots comparing tanh and the convergent.

$$R(x) = \frac{x}{1 + \cfrac{x^2}{3 + \cfrac{x^2}{5 + \cfrac{x^2}{7 + \cfrac{x^2}{9 + \frac{x^2}{11}}}}}}.$$

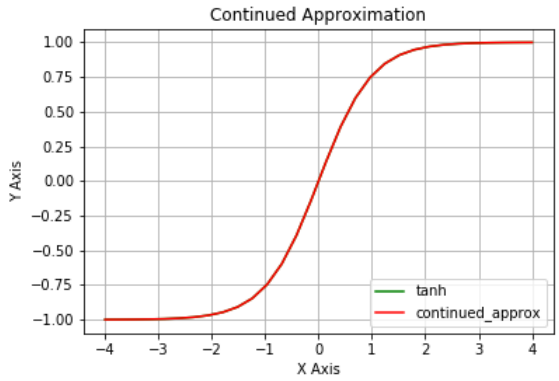


Figure 5.7: Continued Approximation of tanh

We approximate the function on $[-4, 4]$, and the rest part of this function is asymptotic type. Now if Warren result applicable, we use it and Theorem 5.3.1 or we approximate it by polynomial and then use Theorem 5.3.1.

Using Pade Approximation

This idea based on the Pade approximants of $exp(x)$. More precisely, let the (n, n) approximant of $exp(x)$ be represented by $exp(x) \approx \frac{p_n(x)}{p_n(-x)}$, where $p_n(x) = \sum_{j=0}^{n} \frac{\binom{n}{j}}{j!\binom{2n}{j}} x^j$.
From this, we find that we can approximate tanh(x) with a rational function like so:

$$tanh(x) \approx \tau_n(x) = \frac{p_n(x)^2 - p_n(-x)^2}{p_n(x)^2 + p_n(-x)^2}.$$

Consider

$$\tau_3(x) = \frac{x(10 + x^2)(60 + x^2)}{600 + 270x^2 + 11x^4 + \frac{x^6}{24}}.$$

Here are comparison plots for $tanh(x)$ and $\tau_3(x)$:

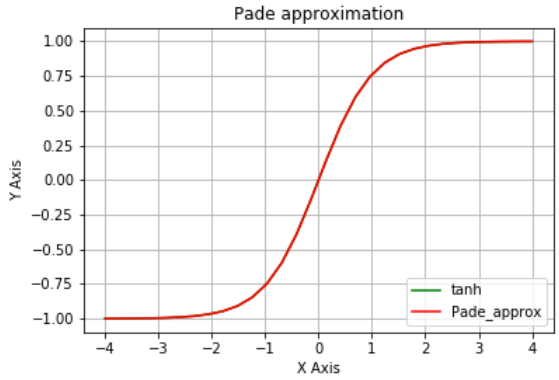


Figure 5.8: Comparison of $tanh(x)$ and $\tau_3(x)$

We approximate the function on $[-4, 4]$, and the rest part of this function is asymptotic type. Now if Warren result applicable, we use it and Theorem 5.3.1 or we approximate it by polynomial and then use Theorem 5.3.1.

5.4 VC Dimension Calculation For RVFLN

5.4.1 First Approach

We consider the network has $u_1, u_2, ..., u_m$ as input nodes value and $a_{i,j}, i \leq m, j \leq n$ are input to hidden layer weights, $b_1, ..., b_n$ are the hidden layer bias, $c_1, ..., c_n$ are hidden to output layer weights, c_0 is output layer bias and output layer has one unit as consider as binary classification problem. Also input to hidden output layer weights are $d_1, ..., d_m$. Then output of the network is

$$y = c_0 + \sum_{i=1}^{m} u_i d_i + \sum_{j=1}^{n} c_j \sigma(A_j u + b_j)$$

By our definition, all $a'_{i,j}s$ are fixed and assigned from a probability distribution. Now the obtained class of functions are span of $1, d_1, ..., d_m, \sigma(A_1 u + b_1), ..., \sigma(A_n u + b_n)$.

So the vector space span by these elements has dimension at most $n + m + 1$. From Lemma 2.3.1, we get $VC(F) \leq n + m + 1$.

5.4.2 Second Approach

From input to hidden unit weights are assigned randomly, no need to tuning these weights during training the network. So, these weights are fix during calculation of VC dimension. RVFLN has one hidden unit and assume binary output. Consider the network has m input units, k hidden units. For a general case class of functions associated with layer i,

$$F^i = F^{(i,1)} \times .... \times F^{(i,d_i)}$$

where d_i is total unit nodes at layer i. Class of function associated with whole network is

$$F = F^l \circ ... \circ F^1$$

where l is total number of layers except input layer. Use this concept on RVFLN, for hidden layer, $F^1 = F^{(1,1)} \times .... \times F^{(1,k)}$, for output layer $F^2 = F^{(2,1)} \times ..... \times F^{(2,L)}$. Now for whole network,

$$F = F^2 \circ F^1 = (F^{(2,1)} \times ..... \times F^{(2,L)}) \circ (F^{(1,1)} \times .... \times F^{(1,k)})$$

Assume first part of the equation is A and second part is B. Then $\Pi_F(m) \leq \Pi_A(m) \times \Pi_B(m)$. From here we get

$$\Pi_F(m) \leq \Pi_{F^{2,1}}(m) \times ..... \times \Pi_{F^{2,L}}(m) \times \Pi_{F^{1,1}}(m) \times ..... \times \Pi_{F^{1,k}}(m)$$

5.5 Comparison of Different VC Dimension Bounds

Already so many VC upper bound have been calculated using different techniques of different type of neural networks. But we can make a study for comparison of the bound for same type activation function classes. This study will give us a concept of bound for different number of parameters and also different input dimension. We hae seen that when input dimension increases, one bound dominant other bound after a certain number of parameters. Here we consider only weights as a parameter not the bias term. Here Layers means all layers except input and output layer. Also each layer has same number of computational units. So total number of computational units is $|Layers| \times |per\ layer\ computatinal\ unit|$. And also total number of parameters is $|computationalunits| \times |number\ of\ incoming\ edge\ to\ each\ unit|$. Total number of operations is $O(|parameters| + |nodes|)$.

The below table for input dimension 10. Assume each hidden layer has same number of unit 10.

Layers	Units	Parameters
2	20	200
5	50	500
10	100	1000
20	200	20000

Table 5.1: For input dimension 10

The following figure give us a idea of different VC upper bound for sigmoidal type networks, piece wise polynomial type networks, piece wise linear type networks.

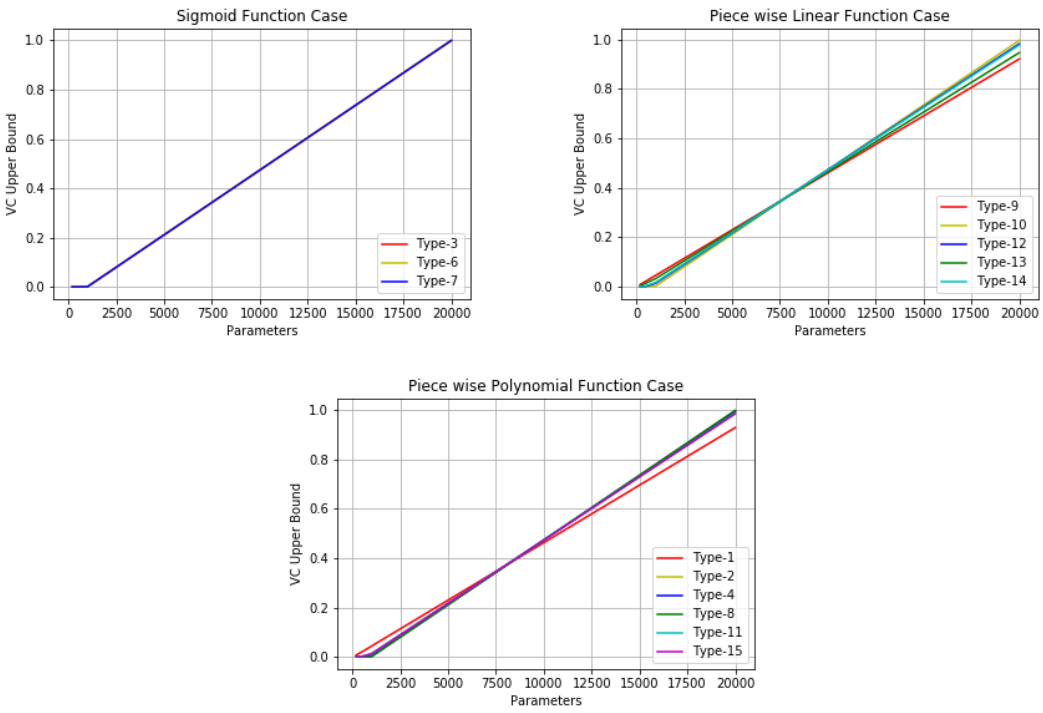


Figure 5.9: For input dimension 10

For sigmoidal neural network all these three bounds are almost same according to our plot. For piece wise linear neural network after a certain number of parameters types bound behaves oppositely with respect to its first part. And for the other types all are almost same. From the figure we have drawn that for piece wise polynomial network type-1 changes its bound value, comparing to the other bounds value after a certain number of parameters. But all the bounds are monotonically increasing. Some of them also are strictly monotonically increasing.

The below table for input dimension is 50. Here each hidden layer has same number of units 30. We assume $p = 1$ and $l = 2, 5, 10, 20$ for layers $= 2, 5, 10, 20$ respectively. If we increase p then bound also change proportionally.

Layers	Units	Parameters
2	60	2400
5	150	5100
10	300	9600
20	600	18600

Table 5.2: For input dimension 50

The following figure give us a idea of different VC upper bound for sigmoidal type networks, piece wise polynomial type networks, piece wise linear type networks.

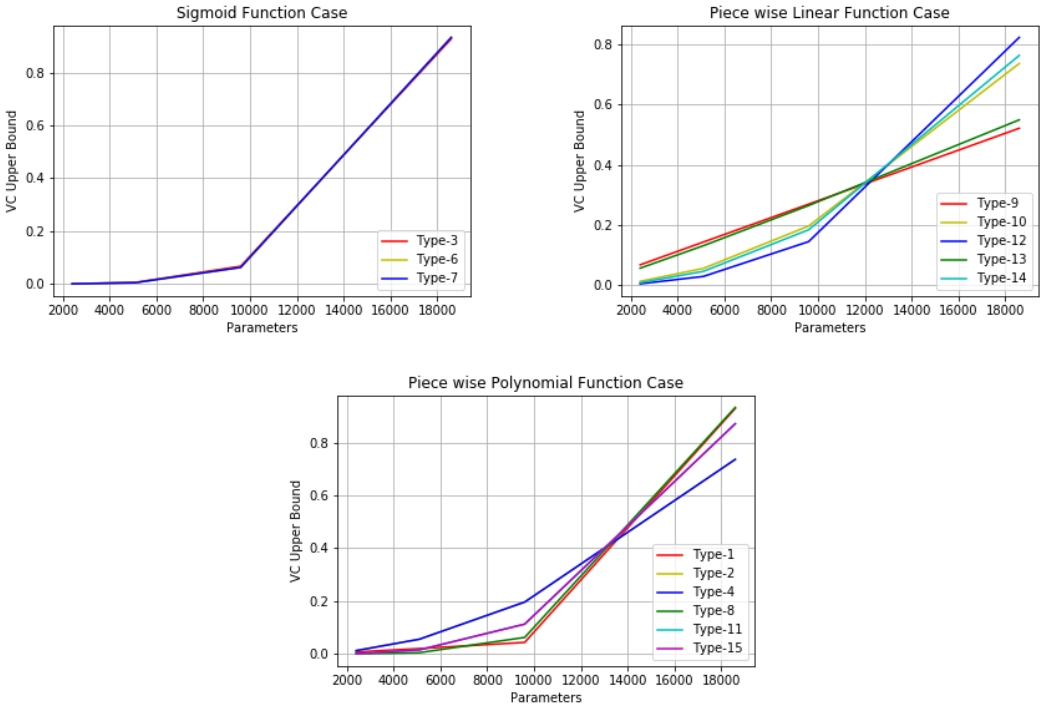


Figure 5.10: For input dimension 50

Here type-i means the function of type i'th row of the main table described later. And also row number starts from 1. Here row $3, 6, 7$ belong so sigmoidal case, row $9, 10, 12, 13, 14$ belong to piece wise polynomial case, row $1, 2, 4, 8, 11, 15$ belong to piece wise polynomial type network.

For sigmoidal neural network all the bounds are basically same but these bounds behaves like strictly monotone increasing function after a certain number of parameters. We can say that for piece wise polynomial neural network, the VC upper bounds behaves differently, means two bounds are strictly monotone increasing function and others are showing this property after a certain number of parameters. Also for piece wise polynomial network, the VC upper bounds crosses one another after a certain number of parameters.

	Constraints	Activation function	Bound	Additional Conditions
Theorem 8.3 of [14]	$F = \{f : \mathrm{R^d} \times X \to \mathrm{R}\}$, $a \in \mathrm{R^d}$,X = input Domain	H - $k-$combination of sgn(F)	$VC(H) \leq 2d\log_2(12kl)$	$m \geq d/k, a \mapsto f(a,x)$ is a polynomial of degree no more than l
Theorem 8.4 of [14]	$h(a,x)$ has no more than t operations,$\mathrm{R^n}$ = input Domain	$h : \mathrm{R^d} \times \mathrm{R^n} \to \{0,1\}$,$H = \{x \mapsto h(a,x) : a \in \mathrm{R^d}\}$	$VC(H) \leq 4d(t+2)$	operations of types: output $0, 1; +, -, *, /; <, >, =, \leq, \geq, \neq$
Theorem 8.14 of [14]	$h(a,x)$ has no more than t operations, $\mathrm{R^n}$ = input Domain	$h : \mathrm{R^d} \times \mathrm{R^n} \to \{0,1\}$,$H = \{x \mapsto h(a,x) : a \in \mathrm{R^d}\}$	$VC(H) \leq t^2d(d+19\log_2(9d))$	operations of types: output $0, 1; +, -, *, /; <, >, =, \leq, \geq, \neq; x \mapsto exp(x)$
Theorem 8.7 of [14]	w weights,k computational units	linear threshold,Piece wise polynomial of p pieces and degree no more than l	$VC(H) = O(w(w + kl\log_2(p)))$	feed forward network,output unit linear threshold
Theorem 8.11 of [14]	w parameters,first layer computation units $= k$, fan in of first layer unit no more than N	first layer with sigmoid activation	$VC(H) \leq 2w\log_2(60ND)$	input domain $X = \{D, ..., -D\}^n$, $D \in \mathrm{N}$,two layer feed forward
Theorem 8.13 of [14]	w parameters, k computation units	linear threshold,sigmoidal function	$VC(H) \leq (wk)^2 + 11wk\log_2(s)$,$s = 18wk^2$	feed forward network, $m \geq w$
[9]	l programmable parameters,k input nodes,m computation nodes	sigmoid function	$VC(A) \leq (ml)(ml-1)/2 + l(2m+1)log(2d) + l(3m+1)log((2m+1)l+1) + l(3m+1)log(2d+1)$	d =maximum degree of polynomial μ, constructing from terms and $VC(A) = O((ml)^2)$
[1]	w parameters,k computation units, L layers	Identity function,Piece wise polynomial of degree at most l and p break points	$VC(sgn(F)) \leq 2wL\log(2ewLpk) + 2wL^2\log(l+1) + 2L$	$k \leq w, L \leq w$ and if L,k $= O(w)$,then $VC(sgn(F)) = O(wL^2 + wL\log_2 L)$
[34]	L layers, p parameters, k computation units	Piece wise constant	$VC(sgn(F)) = O(p)$	Example: Linear Threshold
[16]	L layers, p parameters, k computation units	Piece wise linear	$VC(sgn(F)) = O(p^2)$	Example: ReLu Network
[17]	L layers, p parameters, k computation units	Piece wise polynomial	$VC(sgn(F)) = O(pL^2)$	More Generalized
[27]	w parameters,L layers	piece wise linear	$VC(F) = O(wL\log(w) + wL^2)$	(Bartlett, Maiorov, Meir, 1998)
[27]	w parameters,L layers	piece wise linear	$VC(F) = O(w\log_2(w))$	Cover, 1968; Baum and Haussler, 1989
[17]	W parameters,L layers	ReLu function	$VC(F) = O(WL\log(W))$	Special case of Piece wise Polynomial
[17]	W parameters,U units	piecewise polynomial of degree at most d	$VC(sgn(F)) = O(WU\log(d+1))$	Generalized Case

Table 5.3: Different VC Upper Bounds

5.6 Experiments and Results

Here we will compare test error bound from VC upper bound and practical test error bound. And the experiment has done on the following bench mark datasets 1) D1 (Shuttle Data) 2) D2 (Skin Segmentation data) 3) D3 (Avila Data) 4) D4 (HTRU2 Data) 5) D5 (Default of credit card clients Data) 6) D6 (Online Shoppers Purchasing Intention Data). All data collected from UCI Machine Learning Repository site. E1: Each hidden layers contain 2 nodes, E2: Each hidden layer contains 3 nodes, E3: Each hidden layers contains 4 nodes.

Dataset	E1			E2			E3		
	L1	L2	L3	L1	L2	L3	L1	L2	L3
D1	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001
D2	0.003	0.021	0.019	0.002	0.007	0.007	0.001	0.002	0.001
D3	0.316	0.316	0.316	0.316	0.288	0.328	0.296	0.264	0.278
D4	0.025	0.024	0.025	0.025	0.024	0.022	0.025	0.026	0.025
D5	0.271	0.271	0.271	0.271	0.271	0.271	0.271	0.271	0.271
D6	0.017	0.064	0.151	0.109	0.098	0.112	0.105	0.133	0.108

Table 5.4: Test Error for Neural Network with ReLU Activation

L1: Contains 3 hidden layers. L2: Contains 4 hidden layers. L3: Contains 5 hidden layers. Here train test split is 0.10. Number of epochs used here is 80, batch size is 20. Also as a optimizer we used Adam optimizer. And also as a important point we tried all these experiments in same setup.

Dataset	E1			E2			E3		
	L1	L2	L3	L1	L2	L3	L1	L2	L3
D1	29	35	41	46	58	70	65	85	105
D2	17	23	29	28	40	52	41	61	81
D3	31	37	43	49	61	73	69	89	109
D4	27	33	39	43	55	67	61	81	101
D5	57	63	69	88	100	112	121	141	161
D6	33	39	45	52	64	76	73	93	113

Table 5.5: Total parameters for Neural Network with ReLU Activation

Now we will give a overview of these data sets. Class ratio consider for total (including training and test part) sample and Total sample consider for samples used for training.

- D1: Total sample: 32909, Total features: 9, No categorical features, Class ratio: 34108 : 2458.
- D2: Total sample: 220551, Total features: 3, No categorical features, Class ratio: 194198 : 50859.
- D3: Total sample: 5622, Total features: 10, No categorical features, Class ratio: 4286 : 1961.
- D4: Total sample: 16108, Total features: 8, No categorical features, Class ratio: 16259 : 1639.
- D5: Total sample: 27000, Total features: 23, No categorical features, Class ratio: 23364 : 6636.

- D6: Total sample: 11097, Total features: 11, No categorical features, Class ratio: 10422 : 1908.

The test error rate is

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h}{l}\log(\frac{l}{h}) - \frac{1}{l}\log(\delta)}$$

Now $R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(x_i, \alpha)|$. h is the VC dimension of the machine and $0 \leq \eta \leq 1$, l is the total number of training examples. The following figure gives a details comparison between real test error and practical test error.

For ReLU neural network $VC(F) = O(L_1 W \log_2(pU) + L_1 L \log_2(d))$. Here $d = 1, p = 1$ and $L_1 \approx L$. So modified bound is $VC(F) = O(LW \log_2(pU))$. Here every symbol has similar meaning as symbols in [16]. But the bound which we have used here is $VC(F) \leq L + (\sum_i W_i).\log_2(\log_2(2epR).\log_2(\log_2(2epR)))$

The following figures give us a idea about theoretical test error bound and practical test error bound. We always say that the theoretical bound is more than the practical bound. As the experiment has done in a general setting, so we can conclude that almost all the time this tradeoff happens. Here all used activation functions are ReLU function.

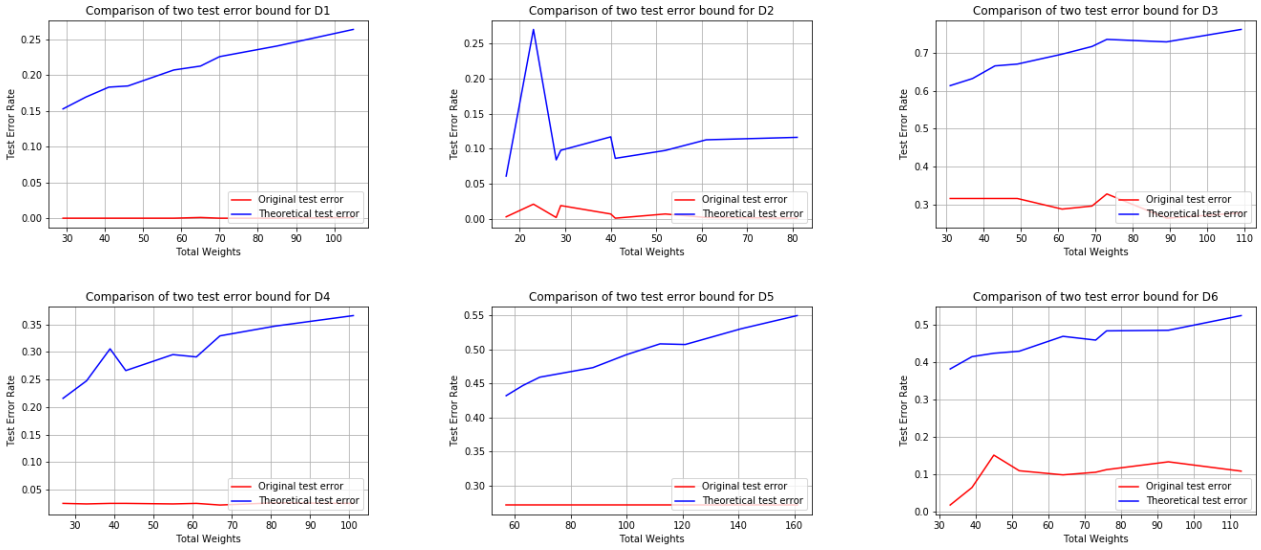


Figure 5.11: For ReLU Neural Network

5.6.1 Conclusion

We know that the Bayesian classifier is the best classifier among all the classifiers. As a conclusion from this statement, we can say that this classifier has the lowest misclassification error. Similarly, We can make a conclusion that the test error bound derived from VC dimension always more than the practical test error bound. From this atleast we get a idea about upper bound of test error rate.

Chapter 6

Future Works

In this chapter we will pose some problems which arises from our proposed methods or solutions. Also we will propose a few things that might have greater importance in statistical learning field.

6.1 Scope of Future Work

Our proposed problems are:

- We have showed that there is a neural network with linear and threshold gates and inputs in $\mathrm{R^m}$ which has VC dimension $O(n^m)$ with weights $O(n^{m-1})$ for $n \geq 1$. Is there some architecture which has same VC dimension bound but has less number of weights?
- Same question as above will arise for neural network with linear, threshold, multiplication, division gates.
- Is there any sufficient method for calculating VC dimension of neural network with skip connections (example RVFLN, ResNet)? Also the upper bound on VC dimension should be tighter than a upper bound of feed forward neural network with out skip connections. We have made this claim because skip connection network gives better performance than without skip connection network in-general.
- Also, there is a result in topology about how many convex regions created by intersecting some number of hyperplanes. Is there similar result for S shape curve? If yes, then we can use that result to calculate VC upper bound for feed forward neural network with S shape functions.
- Besides model theoretic, time taking for each operations, approximation of curves approaches, is there any other way to handle the calculation of VC upper bound for feed forward neural network with sigmoidal activation function with or without skip connections?
- How much the gap between theoretical test error and practical test error depends upon the number of layers, nodes per layers, activation functions in hidden layers?

- Already some researchers have proposed the concept for calculating number of nodes for each hidden layer of a feed forward neural network which depends upon the shape of the data set. For this concept they correlated betti number (number of holes) of this datasets with number of computational nodes. Basically it does not depends upon the activation functions. If we can correlate the VC dimension which depends upon activation function with betti number which correlates with shape of datasets, then we might get more tighter and significant VC bound for each different type of neural networks. And if VC bound depends upon both data sets shape, dimension and activation functions, the bound will be more specific.

Bibliography

[1] Peter L. Bartlett, Vitaly Maiorov, Ron Meir. *Almost Linear VC Dimension Bounds for Piece wise Polynomial Networks.*

[2] P.Koiran, E. D. Sontag. *Neural networks with Quadratic VC Dimension.*

[3] Michael Schmitt. *Lower Bounds on the Complexity of Approximating Continuous Functions by Sigmoidal Neural Networks.*

[4] L. van der Dries. *Tame Topology and o-minimal Structures, University of Illinois.*

[5] J. Knight, A. Pillay and C. Steinhorn. *Definable Sets and Ordered Structures II, American mathematical society.*

[6] M. C. Laskowsky. *VC Classes of Definable Sets, J. London Math Society.*

[7] P. Goldberg, M. Jerrum. *Bounding the VC Dimension of Concept Class Parametrized by Real Numbers.*

[8] A. G. Khovanski, Fewnomials. *American Mathematical Society.*

[9] Marek Karpinski, Angus Macintyre. *Polynomial Bounds for VC Dimension of Sigmoidal Neural Networks.*

[10] M. C. Laskowski. *Vapnik-Chervonenkis Classes of Definable sets.*

[11] H. E. Warren. *Lower Bounds for Approximation by Non linear Manifolds.*

[12] J. Renegar. *On the Computational Complexity and Geometry of the First Order Theory of the Reals.*

[13] Paul W. Goldberg, Mark R. Jerrum. *Bounding the Vapnik Chervonenkis Dimension of Concept Classes Parameterized by Real Numbers.*

[14] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations.*

[15] Marek Karpinski, Angus Macintyre. *Polynomial Bounds for VC Dimension Of Sigmoidal and General Pfaffian Neural Networks.*

[16] Peter L. Bartlett, Nick Harvey, Christopher Liaw, Abbas Mehrabian. *Nearly-tight VC-Dimension and Pseudodimension Bounds for Piecewise Linear Neural Networks.*

[17] Nick Harvey, Christopher Liaw, Abbas Mehrabian. *Nearly-tight VC-Dimension Bounds for Piecewise Linear Neural Networks.*

[18] Wolfgang Maass. *Vapnik-Chervonenkis Dimension of Neural Nets.*

[19] Xu Miao, Lin Liao. *VC Dimension and its Applications in Machine Learning.*

[20] A. J. Wilkie. *Model Completeness Results For Expansions Of The Ordered Field Of Real Numbers By Restricted Pfaffian Functions And The Exponential Function.*

[21] Peter L. Bartlett, Robert C. Williamson. *The VC-Dimension and Pseudodimension of Two-Layer Neural Networks with Discrete Inputs.*

[22] Wee Sun Lee, Peter L. Bartlett, Robert C. Williamson. *Lower Bounds on the VC Dimension of Smoothly Parametrized Function Classes.*

[23] John W. Milnor. *From The Differentiable Viewpoint.*

[24] E. D. Sontag, Angus Macintyre. *Finiteness Result of Sigmoidal Neural Network.*

[25] Martin Anthony and Norman Biggs. *Computational Learning Theory for Artificial Neural Networks.*

[26] Eduardo D. Sontag. *VC Dimension of Neural Networks.*

[27] Peter L. Bartlett, Wolfgang Maass. *Vapnik-Chervonenkis Dimension of Neural Nets.*

[28] Shai Ben David, Nicolo Cesa Bianchi, Philip M. Long. *Characterizations of Learnability for Classes of* $\{0, .., n\}$ *Valued Functions.*

[29] Eshan Chattopadhyay, Pravesh Kothari, Adam Klivans. *An Explicit VC-Theorem for Low-Degree Polynomials.*

[30] Bernard Ycart, Joel Ratsaby. *VC-Dimensions of Random Function Classes.*

[31] Elchanan Mossel and Christopher Umans. *On the Complexity of Approximating the VC Dimension.*

[32] Marek Karpinski, Thorsten Werther. *VC Dimension and Learnability of Sparse Polynomials and Rational Functions.*

[33] Ye Ren, P.N. Suganthan, N. Srikanth, Gehan Amaratunga. *Random Vector Functional Link Network for Short-term Electricity Load Demand Forecasting.*

[34] Baum, Haussler. *What Size Net Gives Best Generalization.*

[35] Peter Bartlett. *CS281B/Stat241B. Statistical Learning Theory.*

[36] Sham Kakade and Ambuj Tewari. *CMSC 35900 (Spring 2008) Learning Theory.*

[37] Mohri, Rostamizadeh, Talwalkar. *Foundations of Machine Learning*