From Analytic to Algebraic: The Algebraic Geometry of Two Layer Neural Networks

Spencer Wong

Supervised by Dr. Daniel Murfet

Thesis for the Masters of Science
May 2022

School of Mathematics and Statistics
The University of Melbourne

Acknowledgements

This thesis would be nothing without the guidance and support of my supervisor Dr. Daniel Murfet. A huge thank you for not only guiding me to a project worth pursuing, but also for the knowledge, advice, and inspiration provided along the way which lie at the heart of this project. A special thank you to my friends, including Izzy for her friendship and support throughout the whole degree, and also for her incredibly detailed proofreading, and Liam for his advice on to getting through the ups and downs of a Master's thesis. Finally, a huge thank you to my family, who've shown nothing but kindness and patience towards my heightened state of anxiety over the last couple of years.

Contents

Chapter 1

Introduction

As neural networks rapidly change scientific discovery and technological development, the simplest and smallest of them remain difficult to rigorously understand. One of the most promising approaches for studying these networks is Sumio Watanabe's Singular Learning Theory (SLT). It combines algebraic geometry and statistics to show that the singular structure of these models makes their generalisation performance theoretically predictable [Wat09]. In this approach, a key but often overlooked step is to transition from the study of analytic functions and analytic sets to the study of polynomials and algebraic sets.

This thesis focuses on the transition from analytic to algebraic in biasless two layer tanh networks, which are functions of the form

$$f(x,w) = \sum_{i=1}^{n} a_i \tanh(b_i x)\,,$$

where $x \in \mathbb{R}$ is the input and $w = (a_1, ..., a_n, b_1, ..., b_n) \in \mathbb{R}^{2n}$ is the parameter. The goal of many machine learning problems is to find a parameter w which makes the network closely match some "true" function, $f_T : \mathbb{R} \to \mathbb{R}$. This "closeness" between the model f and the true function f_T at parameter $w \in \mathbb{R}^{2n}$ is given by the Kullback Leibler (KL) divergence, which in this case can be shown to be

$$K(w) = \frac{1}{2}\int_{\mathbb{R}} (f_T(x) - f(x,w))^2\, q(x)dx\,,$$

where $q(x)$ is the *input distribution*.

Typically $K(w)$ is analytic but not algebraic (ie: a polynomial or rational function) and "transitioning from analytic to algebraic" refers to replacing $K(w)$ with a polynomial that is equivalent for the purposes of SLT. For example, consider the two neuron network $f(x,a,b,c,d) = a\tanh(bx)+c\tanh(dx)$. The KL divergence for this model is given by the integral $\int_{[-1,1]}(a\tanh(bx)+c\tanh(dx))^2dx$, which is analytic. We will see that this can be replaced by the equivalent polynomial

$$(ab+cd)^2 + (ab^3+cd^3)^2\,.$$

This polynomial is simpler and easier to study than $K(w)$. In fact, finding equivalent polynomials is crucial for understanding the behaviour of neural networks in SLT.

What makes two layer neural networks difficult to study, is the fact they are *singular* statistical models. In particular the map $w \mapsto f(-,w)$ is not injective. Such models fail the assumptions of common statistical tools, and hence we require a whole new theory to describe their behavior. Singular Learning Theory was developed to fill this gap.

At the heart of this theory, Watanabe suggests, is a "fundamental relation between algebraic geometry and statistical learning theory" [Wat09]. Algebraic geometry appears in SLT primarily through Hironaka's theorem on the resolution of singularities, which Watanabe uses to show that each model

has an associated invariant called the *Real Log Canonical Threshold (RLCT)*. The RLCT determines a model's ability to generalise to new data, and hence a central way to understand how a model works is to calculate its RLCT.

In general, this is no simple task. A model's RLCT is calculated by resolving the singularities of the KL divergence. This involves applying repeated *blow ups*, a tool from algebraic geometry which simplifies singularities, and while such a desingularisation is theoretically known to exist, it may be difficult to calculate in practice. This is where the transition from analytic functions to algebraic geometry and polynomials comes in. Equivalent polynomials are defined to have the same geometry as the KL divergence, meaning they have the same RLCT. Computing the RLCT may be much easier using a polynomial rather than the original KL divergence, and in practice, every nontrivial calculation of an RLCT in the literature relies on this approach.

While equivalent polynomials are crucial for calculations, the literature does not provide clear methods for finding them. This step is rarely explained in much detail, and the methods used appear difficult to generalise to new models. This can be a major roadblock for both understanding the theory and applying it to new examples.

This thesis aims to clarify this process in two layer tanh networks. In Chapter 2-Chapter 4, we develop general methods to derive equivalent polynomials for neural networks, and apply them to the two layer example, resulting in a new proof that the KL divergence of these networks is bounded by a polynomial (this was proven by other methods in [Wat00c]). We also show a small application of these equivalent polynomials in Chapter 5, where we study their geometry while varying the true function f_T. For a detailed summary of this thesis, see Section 1.0.2.

1.0.1 Explanation of the main methods and results

In our search for methods to replace $K(w)$ with a polynomial, the starting point was a remark in Watanabe's book on SLT, *Algebraic Geometry and Statistical Learning Theory* [Wat09, Remark 7.6]. The remark proposes a general set of conditions based on a model's Taylor series, which it claims automatically give an equivalent polynomial.

> **Remark 7.6** If a statistical model $p(y|x,w)q(x)$ and a true distribution $q(y|x)q(x)$ are respectively given by
>
> $$p(y|x,w) = \frac{1}{2}\exp\left(-\frac{1}{2}(y-f(x,w))^2\right),$$
> $$q(y|x) = \frac{1}{2}\exp\left(-\frac{1}{2}(y-f_T(x))^2\right),$$
>
> ... If
>
> $$f(x,w) - f_T(x) = \sum_{j=0}^{\infty} f_j(w)e_j(x), \tag{1.0.1}$$
>
> where $f_k(w)$ is a set of polynomials and $\{e_k(x)\}$ is a set of linearly independent functions on the support of $q(x)$, then by the Hilbert basis theorem, there exists J such that $K(w)$ is equivalent to
>
> $$K_1(w) = \sum_{j=0}^{J} f_j(w)^2.$$

According to [Wat09, Remark 7.6], all that is needed to find an equivalent polynomial is a Taylor series which is a sum of polynomials multiplied by functions of the input x. With such general assumptions, it has the potential to apply to a wide class of neural networks.

However, [Wat09, Remark 7.6] is not proven in the book, nor, to the best of our knowledge, in the wider literature. We cannot apply it to new examples without first understanding if and how it would

work. As we will discuss in Section 1.1.3, our attempts to prove the remark ultimately failed and it remains unclear if it holds in its stated generality. Notably, Watanabe never directly applies this remark in any examples.

What the remark does provide though, is a promising set of ideas and techniques for finding equivalent polynomials. Decomposing a model into a series of the form in (1.0.1) and using the Hilbert basis theorem make up the core of our approach in this thesis. This approach adds several technical conditions though, and in checking these, we will further explore Watanabe's proposed link between statistical learning theory and algebraic geometry, introducing tools from computational algebraic geometry such as Gröbner bases and modified versions of the polynomial division algorithm.

The main result of this approach is the following theorem, which gives a polynomial upper bound for the KL divergence of two layer tanh networks, even in cases where the true parameter is non-zero.

Theorem. (Main Theorem) Given a biasless two layer neural network with n neurons, f, with input space $X = [-t, t]$ for some $0 < t < \frac{\sqrt{\pi}}{2\sqrt{2}}$ and weight space $W = \mathbb{R}^n \times [-s, s]^n$ where $0 < s < \sqrt{2\pi}$, then for any given true parameter $w_0 = (a_{0,1}, ..., a_{0,n}, b_{0,1}, ..., b_{0,n}) \in W$, the Kullback Leibler divergence satisfies

$$K(w) \leq C \sum_{j=0}^{2n-1} \left(\sum_{l=1}^{n} a_l b_l^{2j+1} - a_{0,l} b_{0,l}^{2j+1} \right)^2 \tag{1.0.2}$$

as functions on W for some constant $C > 0$.

This upper bound was proven by Aoyagi and Watanabe in [AW09, Wat01a]. In fact, their method appears to show both an upper and lower bound using the polynomial. However we found their proof difficult to follow, and so have developed an alternative proof of the upper bound which we hope is easier to understand.

Though proving equivalence also requires a lower bound, an upper bound is still meaningful by itself. Upper bounds on the KL divergence can provide bounds for the RLCT, which in turn give upper bounds on the asymptotic behaviour of a model's generalisation error.

Equivalent polynomials let us explore one of the key conceptual points of SLT – that the geometry of the KL divergence affects the generalisation performance of a model. The polynomials make it easy to find the zero set of $K(w)$. For the tanh network with two neurons, we projected this zero set into 3 dimensions and plotted it, with the result shown in Figure 1.1.

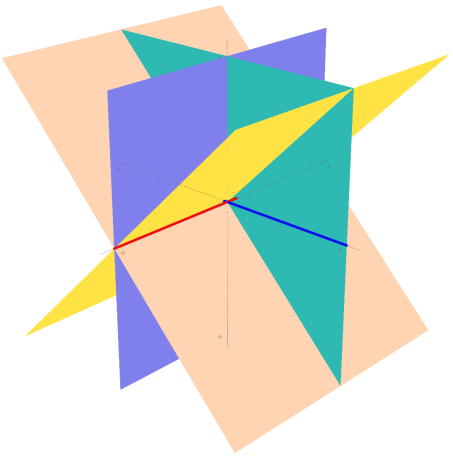


Figure 1.1: The geometry of a two neuron tanh network. For a detailed analysis of this set, see Chapter 5.

1.0.2 Summary of this thesis

While the main focus of this thesis is two layer neural networks, much of the discussion will be more general. In the first three Chapters we examine [Wat09, Remark 7.6] quite generally and develop techniques which might apply to many types of networks. Once we have introduced these methods, we will zero in on two layer biasless tanh networks, using our results to find polynomial upper bounds for their KL divergences.

We will begin by introducing the background of SLT in the first Chapter. This will include all the relevant statistical definitions and an outline of Watanabe's process for calculating RLCTs. The first Chapter will also set up the problem of finding equivalent polynomials, giving a mathematically precise notion of "equivalence" and discussing the challenges of proving [Wat09, Remark 7.6] in its stated generality.

Chapter 2 examines which models satisfy the assumptions of [Wat09, Remark 7.6]. We show that the Taylor series of biasless tanh neural networks have coefficients that are polynomials of the network weights, implying that the remark would in fact apply to these models. Having Taylor series of this form also has implications on the set of true parameters, which we show are not just analytic, but algebraic.

It is unclear if the assumptions in [Wat09, Remark 7.6] alone are enough to guarantee a polynomial upper bound for the KL divergence. In Chapter 3 we find extra conditions which ensure such an upper bound holds. Specifically, we show that if a collection of series made up of the Taylor series polynomials converge and are bounded, then a model's KL divergence is bounded by a polynomial. Checking these conditions for actual models is not straightforward though, and involves dividing a sequence of polynomials by a finite generating set. To carry out this division we introduce methods from computational algebraic geometry such as *Gröbner bases* and the *generic division algorithm*.

In Chapter 4 we apply these methods to two layer neural networks of various sizes. For networks with two neurons, the generic division algorithm and Gröbner bases can successfully be used. However for larger networks these methods increase in complexity and become unworkable. To analyse two layer networks with arbitrarily many neurons, we instead exploit the patterns in their Taylor series polynomials. The results obtained from this technique let us investigate models in which the true parameter is non-zero, leading to the Main Theorem of the thesis.

With the polynomials from Chapter 4, we can analyse the geometry of the KL divergence. In Chapter 5 we compute the irreducible decompositions of the sets of true parameters for small neural networks. These decompositions let us plot the zero sets, and observe how they change as the true function f_T changes. We then use Aoyagi and Watanabe's formula for the RLCT [AW09] to compute the RLCTs for these networks and observe how the value of the RLCT is reflected in the geometry of the zero set.

1.1 Background

1.1.1 Models and regression functions

A common problem in statistics is a *learning problem*, where the goal is to infer some underlying relationship between elements of two spaces using a dataset of known examples. The spaces are called the input space $X \subset \mathbb{R}^N$, and output space $Y \subset \mathbb{R}^M$. The elements of these spaces are assumed to be related by some conditional probability distribution $q(y|x)$ called the *true distribution*. Additionally, an *input distribution* $q(x)$ describes how common each element of the input space is.

Learning problems are set up by constructing a parameterised conditional probability distribution called the *model*, $p(y|x,w)$. The value $w \in W \subset \mathbb{R}^d$ is called the *weight* or *parameter*, and each choice of weight, w, determines a conditional probability distribution via the function $(x,y) \mapsto p(y|x,w)$. The goal of many learning problems is to use a dataset of known input-output pairs (x,y) to find a parameter $w \in W$ making $p(y|x,w)$ match the true distribution $q(y|x)$ as closely as possible.

Along with a model $p(y|x,w)$, true distribution $q(y|x)$, and input distribution $q(x)$, learning problems are equipped with a *prior probability distribution* $\varphi(w)$. The prior is a probability distribution on the set of parameters W which describes one's initial belief on how likely each weight is before any data has been seen.

The "closeness" between the true distribution $q(y|x)$ and the parameterised distribution $p(y|x)$ is measured by the following function.

Definition 1. The *Kullback Leibler (KL) divergence* from the parameterised distribution p to the true distribution q at parameter w is given by

$$K(w) = \int q(y|x)q(x) \log\left(\frac{q(y|x)}{p(y|x,w)}\right) dydx\,. \tag{1.1.1}$$

Remark. The KL divergence has the following properties

- $K(w) \geq 0$ for all w.
- $K(w) = 0$ if and only if $p(y|x,w) = q(y|x)$ almost everywhere with respect to $q(x)dydx$.

For the models considered in this thesis, $K(w)$ is an analytic function.

Learning problems, such as supervised machine learning, are often based around some *regression function* $f : X \to Y$ which takes an input and maps it to a corresponding output. In machine learning, this function f is parameterised by a weight, $w \in W$, and can be thought of as a function from $X \times W \to Y$. The goal of supervised learning tasks is to then use a dataset of known examples to find a weight w so that the function $f(-,w) : X \to Y$ "performs well". In order to bring such a function into the framework of statistical learning, a common technique is to construct a conditional probability distribution by setting

$$p(y|x,w) = \frac{1}{\sqrt{(2\pi)^M}} \exp\left(-\frac{1}{2}\left\|y - f(x,w)\right\|^2\right)\,, \tag{1.1.2}$$

where M is the dimension of the output space (ie: $Y = \mathbb{R}^M$). So given a choice of weight w and input x, the probability of obtaining output y is given by a normal distribution centered at $f(x,w)$ with identity covariance matrix.

In many problems, it is assumed that the true distribution is generated in a similar manner from some *true function* $f_T : X \to Y$, so that

$$q(y|x) = \frac{1}{\sqrt{(2\pi)^M}} \exp\left(-\frac{1}{2}\left\|y - f_T(x)\right\|^2\right)\,. \tag{1.1.3}$$

This true function maps an input to the "correct" output.

Example. A classic machine learning problem is digit recognition where the goal is to find a function which takes an image of a handwritten digit and identifies the digit it depicts. The input space is a set of greyscale images made up of $m \times m$ pixels, each with a brightness described by a number $x \in [0,1]$, where 0 is black and 1 is white. Hence the input space for this example is $[0,1]^{m^2} \subset \mathbb{R}^{m^2}$.

The output space needs to describe the digits $\{0,1,2,3,4,5,6,7,8,9\}$. This is often done using the space $[0,1]^{10}$, where the digit i is encoded as the vector with 0 in all but the $i+1^{\text{th}}$ component, which equals 1.

The true function for this example $f_T : [0,1]^{m^2} \to [0,1]^{10}$, would take an image of a handwritten digit and correctly map it to the digit it depicts.

While the original definition of the KL divergence (1.1.1) can be opaque and hard to interpret, it takes a much simpler form when both the true distribution and model are constructed from regression functions.

Lemma 1. When the true conditional distribution $q(y|x)$ and model $p(y|x,w)$ take the forms given in (1.1.3) and (1.1.2), the Kullback Leibler divergence is

$$K(w) = \frac{1}{2}\int_X \|f_T(x) - f(x,w)\|^2 \, q(x)dx. \tag{1.1.4}$$

In other words, the KL divergence equals the mean squared error between the regression function and the true function, where the average is calculated with respect to the input distribution $q(x)$.

This is proven by Carroll in [Car21, Lemma A.2].

1.1.2 Singular Learning Theory

Definition 2. Given a parameterised model $p(y|x,w)$, a true conditional distribution $q(y|x)$ is *realisable* if there exists $w \in W$ so that $K(w) = 0$. That is, w makes $p(y|x,w)$ into the same conditional probability distribution as $q(y|x)$.

Definition 3. A weight w which makes $K(w) = 0$ is called a *true parameter*. The set of all such weights

$$W_0 = \{w \in W \mid K(w) = 0\},$$

is called the *set of true parameters* or *zero set*.

Watanabe showed that the structure of the set of true parameters greatly influences the behaviour of a model, and this structure differs dramatically based on a model's *Fisher information matrix*.

Definition 4. The *Fisher information matrix* for a model, $p(y|x,w)$, where $X = \mathbb{R}^N$ and $w \in \mathbb{R}^d$ is the $d \times d$ matrix with components

$$I_{jk}(w) = \int_{\mathbb{R}^N} \left(\frac{\partial}{\partial w_j}\log(p(y|x,w))\right)\left(\frac{\partial}{\partial w_k}\log(p(y|x,w))\right) p(y|x,w)q(x)dxdy. \tag{1.1.5}$$

Definition 5. A statistical model $p(y|x,w)$ is called *identifiable* if the function $w \mapsto p(-|-,w)$ is injective.

Definition 6. A model $p(y|x,w)$ is *regular* if it is identifiable and the Fisher information matrix is positive definite for all parameters $w \in W$.

Definition 7. A model $p(y|x,w)$ is called *singular* if it is not regular, that is either it's not identifiable, or its Fisher information matrix is not positive definite at some weight $w \in W$.

Many well known tools in statistics only work for regular models. The KL divergence of a regular model is given locally by a sum of squares and the set of true parameters is always a single point [Wat09]. As a result, the Bayesian posterior distribution asymptotically approaches a normal distribution, a fact which has been used to derive popular model selection tools such as the *Bayes Information Criterion (BIC)* [Wat13].

The KL divergence for singular models on the other hand, can be much more interesting. $K(w)$ for a singular model is often a complicated analytic function which cannot be written as a sum of squares and the set of true parameters can be an analytic set containing singularities. Model selection tools like the BIC are hence inapplicable and a whole new theory is required to understand their behaviour, namely Watanabe's SLT. The theory shows that the structure of the singularities in the set of true parameters dramatically alters the generalisation abilities of a model. The argument for this statement begins by analysing the following *zeta function*.

Definition 8. From the KL divergence, $K(w)$, and prior distribution $\varphi(w)$ for a model, Watanabe defines the complex valued *zeta function* [Wat09]

$$\zeta(z) = \int K(w)^z \varphi(w)dw\,, \tag{1.1.6}$$

where $z \in \mathbb{C}$.

Proposition 1. The zeta function is holomorphic for $Re(z) > 0$ and can be analytically continued to a meromorphic function on $\mathbb{C}$ whose poles are all negative rational numbers.

This proposition is a core part of SLT and provides the most apparent link between algebraic geometry and statistical learning theory. Watanabe uses Hironaka's resolution of singularities to write $K(w)$ locally as a monomial, using this form to read off the largest pole of $\zeta(z)$ [Wat09].

Definition 9. The largest pole of the zeta function for a model is written as $(-\lambda)$. The non-negative rational number λ is called the model's *Real Log Canonical Threshold (RLCT)*.

Watanabe shows that the RLCT controls the asymptotic behavior of a model's free energy and hence generalisation error as the dataset of training examples grows in size [Wat09]. Here, the generalisation error refers to the average KL distance between the *Bayes predictive distribution* and the true distribution, where the average is taken over all datasets of pairs $(x, y) \in X \times Y$ of size n drawn from the true joint distribution $q(x, y) = q(y|x)q(x)$. The generalisation error is denoted $\mathbb{E}_n G(n)$, and Watanabe's results show that its asymptotic behaviour as the dataset size n increases is determined by the following [MWG+20]

$$\mathbb{E}_n G(n) = \frac{\lambda}{n} + o\left(\frac{1}{n}\right) . \tag{1.1.7}$$

Hence, a key way to understand a model's generalisation properties is to calculate its RLCT.

1.1.3 Equivalence and the Replacement Strategy

As discussed, calculating an RLCT can be far from simple, and so an important step is to replace a model's KL divergence with an "equivalent" polynomial. The notion of equivalence comes from the following lemma.

Lemma 2. Consider two positive analytic functions $K(w)$ and $H(w)$. If there exist constants $c_1, c_2 > 0$ such that

$$c_1 H(w) \leq K(w) \leq c_2 H(w) \tag{1.1.8}$$

for all $w \in W$, then $K(w)$ and $H(w)$ are said to be *equivalent* and their RLCTs are equal.

A proof of this lemma is given by Waring in [War21, Corollary 3.17].

Rather than trying to resolve the singularities of the analytic KL divergence directly, Lemma 2 implies that if we can find an equivalent polynomial, resolving the polynomial's singularities will give the same RLCT. Finding a resolution for the polynomial may be much easier than for a given analytic $K(w)$.

In his book, Watanabe proposes [Wat09, Remark 7.6] as a method for finding these equivalent polynomials. This remark is left unproven though, and at first glance it can appear trivial. However, trying to prove this remark has so far been unsuccessful, and has led us to many dead ends. To understand why the remark both intuitively makes sense, but is difficult to prove, we will first run through some arguments which are obvious to try but ultimately fail. The lower bound in particular, $c\sum_{j=0} f_j(w)^2 \leq K(w)$, caused us many problems.

Our first thought was to consider everything taking place in the Hilbert space $L^2(X, q)$ which has inner product given by $(f, g) = \int_X f(x)g(x)q(x)dx$. The KL divergence for a model described in the remark is then

$$K(w) = \|f(-, w) - f_T\|^2_{L^2(X,q)} .$$

If you assume that the $e_j(x)$'s are not just linearly independent, but are also orthonormal with respect to the inner product, it immediately follows from (1.0.1) that

$$K(w) = \sum_{j=0}^{\infty} f_j(w)^2 ,$$

and the lower bound $\sum_{j=0}^{J} f_j(w)^2 \leq K(w)$ clearly holds. In no example we've seen though are the $e_j(x)$'s actually orthonormal (unless you cook one up to have this property). The next obvious step to try is to orthonormalise the $e_j(x)$'s using the Gram Schmidt algorithm, which defines

$$\begin{aligned}
\tilde{c}_1 &= e_1\,, \\
\tilde{c}_2 &= e_2 - \frac{(e_2, \tilde{c}_1)}{(\tilde{c}_1, \tilde{c}_1)}\tilde{c}_1\,, \\
&\vdots \\
\tilde{c}_k &= e_k - \sum_{i=1}^{k-1} \frac{(e_k, \tilde{c}_i)}{(\tilde{c}_i, \tilde{c}_i)}\tilde{c}_i\,, \\
&\vdots
\end{aligned}$$

before normalising by setting $c_i = \frac{\tilde{c}_i}{\|\tilde{c}_i\|}$.

We could then write $f(x,w) - f_t(x) = \sum_{j=0}^{\infty} g_j(w)c_j(x)$ where $g_j(w) = (f(-,w), c_j)$ and it would certainly be true that

$$K(w) = \sum_{j=0}^{\infty} g_j(w)^2\,.$$

The problem with this approach is that the $g_j(w)$'s aren't necessarily polynomials. For example (c_1, e_j) could be non-zero for infinitely many j (since $c_1 \propto e_1$, and the $e_j(x)$'s aren't orthogonal), meaning $g_1(w)$ would be an infinite sum of the $f_j(w)$'s and potentially a power series rather than a polynomial. This technique would then fail to produce an equivalent polynomial.

Another approach was to consider the linear function $\ell^2(\mathbb{R}) \to L^2(X,q)$ sending $(\alpha_j)_{j=0}^{\infty} \mapsto \sum_{j=0}^{\infty} \alpha_j e_j(x)$. If this function was well defined, injective, and bounded bellow, and if $(f_j(w))_{j=0}^{\infty} \in \ell^2(\mathbb{R})$ for all values of $w \in W$, the lower bound would follow. Checking injectivity and boundedness however is in general not at all simple, and our work on proving the lower bound has stalled for the moment.

Hence, we instead decided to focus on understanding the upper bound half of the equivalence condition, $K(w) \leq c_2 \sum_{j=0}^{J} f_j(w)^2$. Even though such a bound is only half of the equivalence requirement, finding an upper bound is still meaningful in itself. Lemma 2 is a corollary of the following result:

Lemma 3. If there exists a real number $c > 0$ such that $K(w) \leq cH(w)$ then the RLCT for $K(w)$ is less than or equal to the RLCT of $H(w)$.

Hence finding a polynomial upper bound can help find a bound on a model's RLCT. Bounding the RLCT is informative, as doing so provides a bound on the asymptotic behaviour of a model's generalisation performance (as in (1.1.7)).

The general strategy we will use for finding polynomial upper bounds is based on [Wat09, Remark 7.6]. This strategy will be referred to as the *Replacement Strategy* and involves the following steps:

Replacement Strategy

1. Check the Taylor series of a model and true function has the form

$$f(x,w) - f_T(x) = \sum_{j=0}^{\infty} f_j(w)e_j(x)\,,$$

   where the $f_j(w)$'s are polynomials. We will frequently refer to this as the *condition of the Replacement Strategy*.

2. Find a finite basis $\{f_0, f_1, ..., f_J\}$ for the ideal $I = \langle \{f_j\}_{j=0}^{\infty} \rangle$.

3. Do some algebra to show $K(w) \leq C \sum_{j=0}^{J} f_j(w)^2$.

Remark. While [Wat09, Remark 7.6] requires the functions of the input, the $e_j(x)$'s, to be linearly independent on $\text{supp}\{q(x)\}$, we have left this condition out of the Replacement Strategy. In Section 3.1, we will see that this condition is not necessary for finding an upper bound.

1.2 A history of two layer tanh networks and equivalent polynomials in Singular Learning Theory

The development of SLT and the study of two layer tanh networks have gone hand in hand from the beginning. Neural networks were one of the earlier types of models whose non-regularity was noticed, and the failure of statistical techniques like the BIC to describe them appears to have been a major motivation for the development of SLT. As this development occurred and the theory became more sophisticated, it revealed deeper insights into the behaviour of two layer networks.

Before Watanabe began working on SLT, Hagiwara, Toda, and Usui realised in 1993 that the Akaike Information Criterion could not be derived for two layer neural networks because of the non-uniqueness of each weight [HTU93]. Fukumizu further examined the non regularity of neural networks. In 1996, he studied which parameters w make the Fisher information matrix singular for a two layer neural network, $f(x, w)$. He found that this matrix $I(w)$ is positive definite at a parameter w if and only if the function $f(-, w)$ does not equal that of a neural network with less neurons [Fuk96].

Noting this failure of the regularity condition and model selection tools in layered neural networks, Watanabe began developing new techniques to study their generalisation behaviour. In the paper *On the Generalization Error by a Layered Statistical Model with Bayesian Estimation* from 1998, he bounded the generalisation error of a two layer neural network by comparing individual terms in the KL divergence with polynomials [Wat00c].

While polynomials played an important role in this analysis, the approach was quite different to the techniques Watanabe would later develop. The analysis was very specific to the example of two layer tanh networks, and didn't use zeta functions, RLCTs, or blow ups. Additionally, the bounds obtained in this paper are "coarser" than those he found later.

After this result, Watanabe started developing the main tools of SLT, defining the zeta function and RLCT, and showing how they provide a bound on a model's generalisation error in 1999 [Wat99a]. The main example of a singular model used to motivate these techniques was the two layer tanh network. However at this time, Watanabe was not yet able to compute RLCTs for these models.

It was not until Watanabe started replacing the KL divergence with polynomials that he was able to compute the RLCTs exactly for neural networks. The first example of this replacement is in the 1999 paper *Algebraic Analysis for Singular Statistical Estimation*, which looks at the two neuron network $f(x, a, b, c, d) = a\tanh(bx) + c\tanh(dx)$ [Wat99c]. Here Watanabe found that the RLCT is $\lambda = \frac{2}{3}$ by replacing the model's KL divergence with the polynomial $(ab+cd)^2 + (ab^3+cd^3)^2$ and then computing blow ups. The paper doesn't justify the replacement in much detail, with no mention of equivalence. Instead, it cites [Wat00c] to explain this step.

The same calculation for the two neuron network appears in several other papers from the same time period. All of these papers replace the model's KL divergence with the same polynomial, and again cite [Wat00c] to explain the replacement [Wat99b, Wat00b].

Having found the RLCT of networks with two neurons, Watanabe set out to understand larger networks, attempting the problem for networks with arbitrarily many neurons. The first few papers studying larger networks didn't actually replace the KL divergence with a polynomial, and as a result only found bounds on the RLCT. Here, Watanabe factored some the weights out of the integral defining $K(w)$, and used this factorisation to partially resolve the singularities of K [Wat00a, Wat01b]. The partial resolution let Watanabe find some, but not all of the poles of the zeta function. Since the RLCT is given by the negative of the largest pole of the zeta function, finding any poles gives a bound on the RLCT.

It would be some time before the exact RLCTs of general two layer tanh networks would be found. In the mean time, Watanabe and his collaborators spent a few years using the techniques honed on tanh networks to explore other types of singular models. For example, Yamazaki and Watanabe repeated the technique of factorising the KL divergence and finding a partial resolution to bound the RLCTs of hidden Markov models and mixture models [YW03b, YW03a]. Meanwhile, Aoyagi and Watanabe were able to completely calculate the RLCTs of reduced rank regression models [AW04]. What made this calculation possible though, was the fact that the KL divergence for a reduced rank regression model is a polynomial.

Eventually in 2005, Aoyagi and Watanabe returned to the problem of finding the RLCT for arbitrarily large two layer tanh networks. First, an intricate calculation found that the RLCT for the network with p neurons, $f(x, a_1, b_1, ..., a_p, b_p) = \sum_{k=1}^{p} a_k \tanh(b_k x)$, with true function $f_T(x) = 0$, is given by

$$\lambda = \frac{p^2 + i^2 + i}{4i + 2},$$

where i is the largest integer satisfying $i^2 \leq p$ [AW05]. In this example, they replaced the KL divergence with the polynomial

$$\sum_{j=0}^{p-1} \left(\sum_{k=1}^{p} a_k b_k^{2j+1} \right)^2,$$

and computed the resolution for this polynomial.

The most general result for the RLCT of two layer tanh networks appeared that same year in the paper [AW09]. Aoyagi and Watanabe computed the RLCT for two layer networks with arbitrarily many neurons in the case where the true function isn't just zero, but is given by another two layer network. They found that the RLCT depended on the value of the true parameter, and jumped to different values based on how many components of the true parameter were zero. To calculate this RLCT, they replace $K(w)$ with the polynomial

$$\sum_{j=0}^{P} \left(\sum_{i=1}^{p} a_i b_i^{2j+1} - a_i^* b_i^{*2j+1} \right)^2,$$

"for sufficiently large P", where $(a_1^*, b_1^*, ..., a_p^*, b_p^*)$ is the true parameter. Even with this replacement, the calculation is still very complicated, taking about 60 pages.

In 2009, Watanabe compiled his work on Singular Learning Theory into the book *Algebraic Geometry and Statistical Learning Theory* [Wat09], which outlines all the main ideas and examples of the theory. Here he introduces [Wat09, Remark 7.6], but as discussed, does not prove it.

A pattern emerges from all these examples. The only models whose RLCTs have been exactly computed are those whose KL divergence either equals a polynomial, or is equivalent to a polynomial. This emphasises the role of finding equivalent polynomials in SLT, showing how necessary it is to have good methods to find them.

A few questions arise from this history of tanh networks in SLT. Firstly, since Aoyagi and Watanabe have already computed the RLCT for two layer networks, even when the true parameter is non-zero, why study these networks any more? Hasn't everything you'd hope to find out already been discovered? While the results themselves are very important, we're most interested in understanding the methods used to obtain them, especially the step of replacing the KL divergence with a polynomial. Understanding this is a prerequisite for computing the RLCTs of deeper neural networks, something which so far has not been done.

Secondly, many of the papers which replace the KL divergence with a polynomial cite the paper [Wat00c] for justification, so why haven't we just used the method in that paper? The first reason is that it is difficult to use the paper alone to reconstruct an argument for polynomial equivalence. It wasn't until we neared completing this thesis that we were able to figure out how the arguments the

paper [Wat00c] can prove an upper bound, and we were only able to do so by bringing in the methods we had developed for this thesis. Additionally, the argument in [Wat00c] is very specific to two layer tanh networks, and doesn't look generalisable to other models. It doesn't include any of the ideas in [Wat09, Remark 7.6], which to us, looks like the most promising starting point for developing a general method.

Chapter 2

Taylor series and polynomials

Before focusing on polynomial upper bounds and two layer tanh networks, we will try to understand [Wat09, Remark 7.6] more generally. In this chapter we study which models satisfy the main condition of the remark and the Replacement Strategy.

The main condition requires that the model's regression function and true function can be written as a series

$$f(x,w) - f_T(x) = \sum_{j=0}^{\infty} f_j(w) e_j(x)\,,$$

where the functions of w, $f_j(w)$ are all polynomials. The most obvious way to try and write $f(x,w) - f_T(x)$ as a series of this form is to take its Taylor series in x about the origin $x = 0$, and see whether the coefficients are polynomials as functions of w.

The main result of this chapter is Lemma 5, which shows that the Taylor series of biasless tanh feedforward neural networks always have the above form. We will first introduce what a feedforward neural network is, before proving this lemma via induction on the network depth. We will then show that the sets of true parameters for such models are always algebraic sets.

2.1 Feedforward neural networks

The simplest form of neural networks are *feedforward neural networks*, whose basic structure is used as a starting point for the more sophisticated models in use today. We have chosen to study neural networks rather than other statistical models due to their success in a huge array of complex computational problems from computer vision, language generation, and protein structure prediction, to green tea classification [KSH12, BMR+20, JEP+21, YG21].

Definition 10. A *feedforward neural network* is a function $f : \mathbb{R}^M \to \mathbb{R}^N$ constructed as a composition of functions $f^{(l)} : \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$ called "layers". Each $f^{(l)}$ is an affine transformation followed by a nonlinear *activation function* $\phi : \mathbb{R} \to \mathbb{R}$ applied to each component.

$$\begin{aligned} f^{(l)} &: \mathbb{R}^{d_l} \longrightarrow \mathbb{R}^{d_{l+1}} \\ x &\longmapsto \phi\left(w^{(l+1)}x + b^{(l+1)}\right)\,, \end{aligned}$$

where $w^{(l+1)}$ is a $d_{l+1} \times d_l$ matrix of real numbers called *weights* and $b^{(l+1)} \in \mathbb{R}^{d_{l+1}}$ is a vector of "biases". In the above expression, if $v \in \mathbb{R}^{d_l}$ is a vector, $\phi(v)$ refers to the function ϕ applied component wise to v. The neural network f is then the function

$$f = f^{(n)} \circ f^{(n-1)} \circ \cdots \circ f^{(1)}\,. \tag{2.1.1}$$

The weights $w^{(l)}$ and biases $b^{(l)}$ make up the parameters of a network which are tuned during the learning process. The network described in (2.1.1) is said to have n layers, where the first layer is the input layer, and final layer is called the output layer. The *width* of layer l is d_l, referring to the dimension of its output space $\mathbb{R}^{d_l}$.

A convenient way to visualise a neural network is to think of it as a graph of *neurons* representing *weighted inputs* and *activations* organised in the manner shown in Figure 2.1.

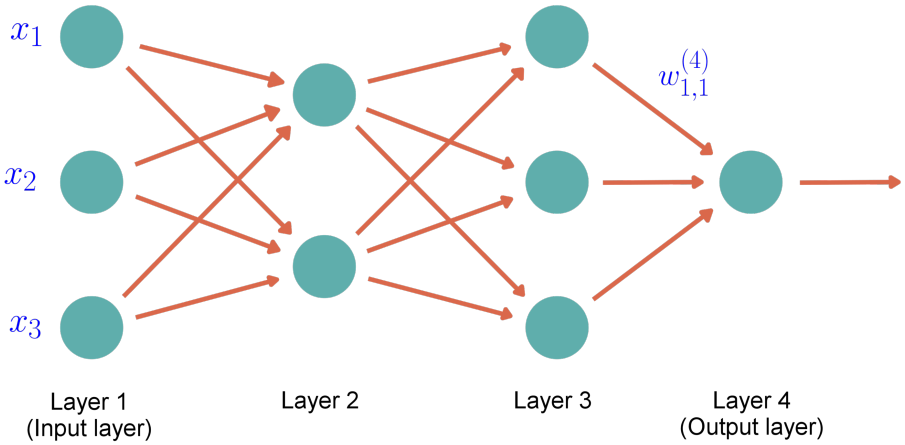


Figure 2.1: A feedforward neural network with four layers. The input layer has three neurons, the second layer has two, the third layer has three, and the output layer has one. This network represents a function from $\mathbb{R}^3$ to $\mathbb{R}$. The weight connecting the first neuron in the third layer to the first neuron in the final layer $w^{(4)}_{1,1}$ has been highlighted.

The output of the l^{th} layer is called the l^{th} activation, denoted by the vector $a^{(l)}(x) = (f^{(l)} \circ ... \circ f^{(1)})(x)$, with the first layer's activation being the input $a^{(1)}(x) = x$. The quantity $z^{(l)}(x) = w^{(l)}a^{(l-1)}(x) + b^{(l)}$ is called the l^{th} layer's weighted input. The network can then be thought of as the following sequence of activations and weighted inputs

$$\begin{aligned} a^{(1)} &= x\,, \\ z^{(2)} &= w^{(2)}(a^{(1)}) + b^{(2)}\,, \\ a^{(2)} &= \phi(z^{(2)})\,, \\ &\vdots \\ z^{(n)} &= w^{(n)}a^{(n-1)} + b^{(n)}\,, \\ a^{(n)} &= \phi(z^{(n)})\,, \end{aligned}$$

with the final network output being given by $f(x) = a^{(n)}(x)$.

The neurons or nodes in Figure 2.1 represent the components of each activation vector. The edges between neurons represent the weight matrices $w^{(2)}, w^{(3)}, ..., w^{(n)}$, with the weight $w^{(l)}_{jk}$ being represented as the edge connecting the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer.

The network in Figure 2.1 is an example of a 1D network, a neural network whose output is a single real number. In this thesis we will only look at 1D networks. While this looks like a significant restriction, multidimensional networks are simple generalisations of the 1D case since each output neuron in a multidimensional network can be thought of as an individual 1D network.

Suppose an n layer multidimensional network $F : \mathbb{R}^M \times W \to \mathbb{R}^N$ has N output neurons. The activations of these output neurons can each be viewed as functions $g_i : \mathbb{R}^M \times W \to \mathbb{R}$ $(i = 1, ..., N)$,

and each of these functions is a 1D neural network. If the true function is given by

$$\begin{aligned} F_T : \mathbb{R}^M &\to \mathbb{R}^N \\ x &\mapsto \left(g_{T,1}(x), ..., g_{T,N}(x)\right), \end{aligned}$$

then the KL divergence for this model and true function is

$$\begin{aligned} K(w) &= \frac{1}{2}\int_{\mathbb{R}^M} \|F(x,w) - F_T(x)\|^2 q(x)dx \\ &= \frac{1}{2}\sum_{i=1}^{N}\int_{\mathbb{R}^M} \left(g_i(x,w) - g_{T,i}(x)\right)^2 q(x)dx\,, \end{aligned}$$

which we can interpret as a sum of N separate KL divergences for several 1D models. If we are able to find equivalent polynomials for the KL divergences of each of the 1D networks, we can add them up to find an equivalent polynomial for the KL divergence of the original N dimensional model.

As well as restricting to 1D networks, we will only study networks with no biases. This restriction is less benign though, since the bias terms $b^{(l)}$ are important in practical networks and are necessary for their universal approximation abilities [LLPS93]. We will see that when biases are included in a network, its Taylor series coefficients will not be polynomials of the weights and we will be unable to study them using the Replacement Strategy. While this limits the applicability of our results, it is still a useful starting point for more work.

2.2 Polynomial coefficient lemma

In this section we show that biasless networks satisfy the condition in the Replacement Strategy when their activation function ϕ has certain properties. In particular, we will see that the Taylor series coefficients for these networks are polynomials of the network weights. To do this, we will use the following definitions.

Definition 11. Let $X = \{x_1, x_2, ...\}$ be a countably infinite set. Define $\mathcal{R} = \mathbb{R}[X]$ to be the real polynomial ring over X, which we define as having a basis of monomials $X^\alpha = \Pi_{i\in\mathbb{N}} x_i^{\alpha_i}$ where α is a sequence of non-negative integers with $\alpha_i \neq 0$ for only finitely many i.

Definition 12. Given a sequence of functions $(g_i : \mathbb{R}^k \to \mathbb{R})_{i\in\mathbb{N}}$, we say that a function $G : \mathbb{R}^k \to \mathbb{R}$ *is a polynomial of the g_i's* if there is a polynomial $p(X) \in \mathbb{R}[X]$ such that as functions, $G = p(g_1, g_2, g_3, ...)$.

Definition 13. Given a feedforward neural network with one dimensional output, $F : \mathbb{R}^m \times W \to \mathbb{R}$, we say that a function $H : \mathbb{R}^m \times W \to \mathbb{R}$ *is a P-function with respect to* F if it is a polynomial of the network weights, the network output F, and the partial derivatives of the second last layer's neuron activations $\frac{\partial^{|\alpha|} a_j^{(n-1)}}{\partial x^\alpha}$ (Where $\alpha \in \mathbb{Z}^m_{\geq 0}$ is a multi-index). In this definition, we can view a weight $w_{kj}^{(i)}$ as the function $\mathbb{R}^m \times W \to \mathbb{R}$ given by the projection of $x \times w$ onto that component $w_{kj}^{(i)}$.

Example: Consider the three layer network F shown in Figure 2.2.

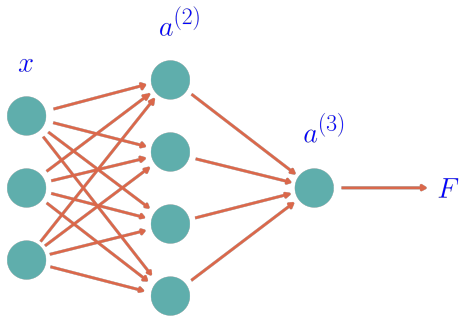


Figure 2.2: A three layer neural network, with three input neurons, four neurons in the second layer, and a single output neuron.

The function

$$G = 2w_{3,1}^{(2)}F^2\left(\frac{\partial^5 a_3^{(2)}}{\partial x_1^2\partial x_2\partial x_3^2}\right)^3 - (w_{11}^{(3)})^5 w_{2,2}^{(2)}\left(\frac{\partial^2 a_1^{(2)}}{\partial x_2\partial x_3}\right)\left(\frac{\partial a_4^{(2)}}{\partial x_1}\right),$$

is a P-function with respect to F.

Lemma 4. Let $F : \mathbb{R}^m \times W \to \mathbb{R}$ be a biasless n-layer neural network with activation function $\phi : \mathbb{R} \to \mathbb{R}$ which satisfies the two conditions

1. $\phi(0) = 0$
2. $\frac{d\phi}{dx} = Q \circ \phi$ for some polynomial $Q \in \mathbb{R}[y]$.

Then if $G : \mathbb{R}^m \times W \to \mathbb{R}$ is a P-function for F, any partial derivative of G with respect to an input variable $\frac{\partial G}{\partial x_i}$ is also a P-function for F.

Proof: First assume that F has k neurons in its second last layer, so that

$$F(x, w) = \phi\left(\sum_{j=1}^{k} w_{1,j}^{(n)} a_j^{(n-1)}\right).$$

It is sufficient to prove the statement for a "monomial" of the form

$$F^n \prod_{l=1}^{r} f_l,$$

where each f_l is a partial derivative (of any order) of a second last layer activation $a_j^{(n-1)}$, and we allow repeated factors. We claim that a partial derivative of this monomial is a P-function with respect to F. The derivative of this monomial with respect to an input x_i is

$$\begin{aligned}
\frac{\partial}{\partial x_i}\left(F^n \prod_{l=1}^{r} f_l\right) &= nF^{n-1}\frac{\partial F}{\partial x_i}\prod_{l=1}^{r} f_l + F^n \sum_{l=1}^{r}\frac{\partial f_l}{\partial x_i}\prod_{j\neq l} f_j \\
&= nF^{n-1}\frac{\partial}{\partial x_i}\left(\phi\left(\sum_{j=1}^{k} w_{1,j}^{(n)} a_j^{(n-1)}\right)\right)\prod_{l=1}^{r} f_l + F^n \sum_{l=1}^{r}\frac{\partial f_l}{\partial x_i}\prod_{j\neq l} f_j \\
&= nF^{n-1}Q(F)\left(\sum_{j=1}^{k} w_{1,j}^{(n)}\frac{\partial a_j^{(n-1)}}{\partial x_i}\right)\prod_{l=1}^{r} f_l + F^n \sum_{l=1}^{r}\frac{\partial f_l}{\partial x_i}\prod_{j\neq l} f_j . \qquad (2.2.1)
\end{aligned}$$

The derivatives $\frac{\partial f_l}{\partial x_i}$ are still partial derivatives of $(n-1)^{\text{th}}$ layer activations, and so the expression in (2.2.1) is a P-function of F. Any P-function for F will be a finite sum of such monomials multiplied by polynomials of the weights. Hence a partial derivative of any P-function for F will also be a P-function for F. □

We are now able to prove the main result of this Section.

Lemma 5 (Polynomial coefficient lemma)**.** Let $\phi : \mathbb{R} \to \mathbb{R}$ satisfy the conditions of Lemma 4. Given a biasless neural network $F : \mathbb{R}^m \times W \to \mathbb{R}$ which has ϕ as its activation function, every partial derivative of F with respect to the inputs x evaluated at $x = 0$ is a polynomial of the weights.

Proof: The main idea is to use induction on the number of layers in the network, with a two layer network as the base case.

During the proof we will frequently use "networks" to refer to 1D biasless neural networks whose activation ϕ satisfies the conditions of Lemma 4.

Base case: A two layer neural network will be of the form

$$F(x, w) = \phi\left(\sum_{i=1}^{m} w_{1i}^{(1)} x_i\right). \tag{2.2.2}$$

Since F is a P-function of itself, by Lemma 4 every partial derivative of F is too. In particular any partial derivative is a finite linear combination of terms of the form

$$w^\alpha F^j \prod_{l=1}^{r} f_l\,,$$

where α is a multi-index and each f_l is a partial derivative of first layer activation. The first layer activations are just the inputs, $a_j^{(1)} = x_j$, so these partial derivatives either equal an input component, or are zero or 1. Hence any partial derivative of F is a linear combination of terms of the form

$$w^\alpha F^n x^\beta\,,$$

where β is another multi-index. By property 1 of Lemma 4, $F(0) = 0$, and it follows that every partial derivative of F in x, evaluated at $x = 0$, is a polynomial of the weights (note that it can be non-zero if $n = 0$, $\beta = 0$).

Induction case: Suppose the claim holds for all networks with n layers. Let F be a network with $n+1$ layers. Then F is itself a P-function for F and so any partial derivative $\frac{\partial^{|\alpha|} F}{\partial x^\alpha}$ is also a P-function for F, meaning it will be a finite sum of terms of the form

$$w^\beta F^t \prod_{l=1}^{r} f_l\,, \tag{2.2.3}$$

where β is a multi-index, $t \in \mathbb{Z}_{\geq 0}$ and each f_l is a partial derivative of an n^{th} layer neuron activation (where we allow repeated factors in the product). Each f_l equals a derivative of the form $\frac{\partial^{|\gamma|} a_j^{(n)}}{\partial x^\gamma}$. However the function

$$\begin{aligned} a_j^{(n)} &: \mathbb{R}^m \times W \to \mathbb{R} \\ (x, w) &\mapsto a_j^{(n)}(x, w)\,, \end{aligned}$$

can itself be thought of as the output of an n layer network. By the induction hypothesis, $f_l(0, w) = \frac{\partial^{|\gamma|} a_j^{(n)}}{\partial x^\gamma}(0, w)$ is then a polynomial of the weights. Using property 1 from Lemma 4 of the activation ϕ, $F(0, w)^t = 0$ or 1 depending on the value of t, so

$$\left(w^\beta F^t \prod_{l=1}^{r} f_l\right)(0, w)$$

is a polynomial of the network weights. It follows that $\frac{\partial^{|\alpha|}F}{\partial x^\alpha}(0,w)$ is as well. □

This result implies that for a network F, within the domain of its Taylor series' convergence,

$$F(x,w) = \sum_{\alpha \in \mathbb{Z}^m_{\geq 0}} f_\alpha(w) x^\alpha \,,$$

where each f_α is a polynomial.

Remark. To see why Lemma 5 requires the network to have no biases, consider a simple two layer counter example

$$F(x,a,b) = \phi\left(ax+b\right) \,,$$

where $x \in \mathbb{R}$ is the input and $(a,b) \in \mathbb{R}^2$ is the weight, where in particular b is a bias. Assuming ϕ is analytic but not a polynomial, the first coefficient in the Taylor series of F about $x=0$ is $F(0) = \phi(b)$, which is not a polynomial.

In Appendix A.2 we will return to this problem and describe a method which may give a polynomial upper bound for two layer networks containing biases.

2.3 Tanh networks

In the previous section we found conditions on the activation function of a network which guarantee its Taylor series coefficients are polynomials of the weights. In this section we use these results to show that tanh networks have polynomial Taylor series coefficients, and we find what these coefficents are when the networks have two layers.

The tanh function satisfies both properties of Lemma 4, since

$$\tanh(0) = 0 \,,$$
$$\frac{d}{dx}(\tanh(x)) = 1 - \tanh(x)^2 \,.$$

Hence by Lemma 5, the Taylor series coefficents for biasless tanh networks are polynomials of the weights, and such networks satisfy the condition of the Replacement Strategy. This does have the caveat that the input and weight spaces, X and W, must small enough for these Taylor series (about $x=0$) to converge. This restriction will be discussed later on in Chapter 4, but for the moment we assume they are sufficiently small.

If $f : \mathbb{R}^m \times W \to \mathbb{R}$ is a biasless tanh network, we can then write

$$f(x,w) = \sum_{\alpha \in \mathbb{Z}^m_{\geq 0}} f_\alpha(w) x^\alpha \,,$$

where each $f_\alpha(w)$ is a polynomial.

If the true function $f_T(x)$ is analytic and X is also small enough for its Taylor series about 0 to converge, then

$$f_T(x) = \sum_{\alpha \in \mathbb{Z}^m_{\geq 0}} c_\alpha x^\alpha \,,$$

where each $c_\alpha \in \mathbb{R}$. It then follows that

$$f(x,w) - f_T(x) = \sum_{\alpha \in \mathbb{Z}^m_{\geq 0}} (f_\alpha(w) - c_\alpha) e_\alpha(x) \,,$$

where $e_\alpha(x) = x^\alpha$. We can then conclude the statistical model defined by f and f_T will satisfy all conditions of the Replacement Strategy.

Remark. The activation function tanh is not the only one which gives polynomial Taylor series coefficents. In the Appendix A.1, we prove that networks with the Swish activation function have this same property.

Two layer tanh networks

A two layer tanh network with n neurons is given by the function

$$f(x, a_1, b_1, ..., a_n, b_n) = \sum_{i=1}^{n} a_i \tanh(b_i x)\,,$$

where $x \in \mathbb{R}$ is the input and $w = (a_1, b_1, ..., a_n, b_n) \in \mathbb{R}^{2n}$ is the weight.

Remark. Many authors call these "three layer networks". The difference in nomenclature depends on whether you count combining the $\tanh(b_i x)$'s into a linear combination as a seperate layer.

By Lemma 5, the coefficients in the Taylor series of f about $x = 0$ are polynomials of the weight w. It's easy to compute these polynomials using the Taylor series of tanh.

$$\tanh(x) = \sum_{j=0}^{\infty} \frac{2^{2(j+1)}(2^{2(j+1)} - 1)B_{2(j+1)}}{(2(j+1))!} x^{2j+1}\,,$$

where B_n is the n^{th} Bernoulli number. This series has a radius of convergence of $\frac{\pi}{2}$ [LLS12]. From this,

$$f(x, a_1, b_1, ..., a_n, b_n) = \sum_{j=0}^{\infty} \frac{2^{2(j+1)}(2^{2(j+1)} - 1)B_{2(j+1)}}{(2(j+1))!} \left(\sum_{i=1}^{n} a_i b_i^{2j+1} \right) x^{2j+1}\,.$$

Writing $c_j = \frac{2^{2(j+1)}(2^{2(j+1)}-1)B_{2(j+1)}}{(2(j+1))!}$, we can rewrite this as

$$f(x, a_1, b_1, ..., a_n, b_n) = \sum_{j=0}^{\infty} c_j \left(\sum_{i=1}^{n} a_i b_i^{2j+1} \right) x^{2j+1}\,.$$

Setting $f_j(w) = \gamma_j \left(\sum_{i=1}^{n} a_i b_i^{2j+1} \right)$ and $e_j(x) = \eta_j x^{2j+1}$, where γ_j and η_j are any real numbers satisfying $\gamma_j \eta_j = c_j$, the Taylor series for f becomes

$$f(x, w) = \sum_{j=0}^{\infty} f_j(w) e_j(x)\,,$$

which is in the form that the Replacement Strategy requires.

In Chapter 4, we use the above expression to find a polynomial upper bound for the KL divergence of two layer tanh networks.

2.4 Algebraic zero sets

We have just seen that biasless tanh networks satisfy the Replacement Strategy's requirements. While the KL divergence for one of these models is an analytic function, the set of true parameters is actually an algebraic set.

Lemma 6. Suppose a model's regression function $f(x, w)$ is analytic, and that every partial derivative of f with respect to x evaluated at $x = 0$ is a polynomial in the weights. Further assume $q(x) > 0$ on some open set, $A \subset \mathbb{R}^m$, and that the true function $f_T(x)$ is analytic. Then the set of true parameters W_0 is an algebraic set.

Proof: For any fixed $w \in W$, the function $\hat{f}_w : x \mapsto f(x,w) - f_T(x)$ is analytic. By the identity theorem for analytic functions, $\hat{f}_w$ is zero on all of $\mathbb{R}^m$ if and only if it is zero everywhere on A. Hence

$$\begin{aligned}
K(w) = 0 &\implies \int_A (f(x,w) - f_T(x))^2 q(x) dx = 0 \\
&\implies (f(x,w) - f_T(x))^2 q(x) = 0 \qquad \text{for almost all } x \in A \\
&\implies f(x,w) - f_T(x) = 0 \qquad \text{for almost all } x \in A \\
&\implies f(x,w) - f_T(x) = 0 \qquad \text{for all } x \in A, \text{ (as } \hat{f}_w \text{ is continuous)} \\
&\implies f(x,w) - f_T(x) = 0 \qquad \text{for all } x \in \mathbb{R}^m \,.
\end{aligned}$$

In particular this implies $\hat{f}_w$ is zero on some open neighborhood of the origin, so that all its derivatives evaluated at $x = 0$ are zero. We've already shown that these derivatives are the polynomials

$$\left(\frac{\partial^{|\alpha|}}{\partial x^\alpha} \hat{f}_w \right)(0) = f_\alpha(w) - c_\alpha \,.$$

We can conclude that

$$K(w) = 0 \implies f_\alpha(w) - c_\alpha = 0 \,,$$

for all $\alpha \in \mathbb{Z}^m_{\geq 0}$. Conversely, if $f_\alpha(w) - c_\alpha = 0$ for all α, then $\hat{f}_w(x) = 0$ for all x in the domain of convergence for its Taylor series about zero. In turn this implies $\hat{f}_w(x)$ is zero on all of $\mathbb{R}^m$, and hence $K(w)$ is zero. We conclude that

$$\{w \in W \mid K(w) = 0\} = \{w \in W \mid f_\alpha(w) - c_\alpha = 0 \,,\, \forall \alpha \in \mathbb{Z}^m_{\geq 0}\} \,,$$

which is the zero set of a sequence of polynomials, ie: an algebraic set. □

Corollary. Combining Lemma 5 and Lemma 6 tells us that biasless tanh neural networks have algebraic sets of true parameters.

Chapter 3

Polynomial upper bounds

In Chapter 2 we saw that a significant class of neural networks meet the conditions in [Wat09, Remark 7.6] and the Replacement Strategy. In this Chapter, we delve deeper into the techniques from the Replacement Strategy, showing that the Hilbert basis theorem can provide a polynomial upper bound to a regression model's KL divergence. Our approach won't be as "automatic" as the remark suggests. Instead, we will have to manually check the convergence of several series in order to find an upper bound.

3.1 Upper bound conditions

Let $f : X \times W \to \mathbb{R}$ be a 1D regression function, and $f_T : X \to \mathbb{R}$ be the true function. Suppose these functions can be written in the form from the Replacement Strategy, that is

$$f(x, w) - f_T(x) = \sum_{j=1}^{\infty} f_j(w) e_j(x) \,,$$

where the $f_k(w)$'s are polynomials.

The Hilbert basis theorem says that every ascending chain of polynomial ideals stabilises. This implies there is some $J > 0$ so that $\{f_0, ..., f_J\}$ generates the ideal $I = \langle \{f_j\}_{j=0}^{\infty} \rangle$ in $\mathbb{R}[w]$, meaning for each j, we can write

$$f_j(w) = \sum_{k=0}^{J} a_j^k(w) f_k(w) \,,$$

where each $a_j^k(w) \in \mathbb{R}[w]$ is a polynomial called a *quotient*. In general these quotients will not be unique [CLO15, p.83].

The following lemma proves that the sum of squares of the above generating set $\sum_{j=0}^{J} f_j(w)^2$ bounds the KL divergence $K(w)$ provided that several series made up of the f_j polynomials and quotients a_j^k converge and are bounded. In the lemma, we use the Hilbert space $L^2(X, q)$ which we defined in Section 1.1.3, as well as the Hilbert space $\ell^2(\mathbb{R})$.

Lemma 7. In addition to the condition of the Replacement Strategy, assume that $(f_j(w))_{j=0}^{\infty} \in \ell^2(\mathbb{R})$ for all $w \in W$, and that the linear map $\tau : \ell^2(\mathbb{R}) \to L^2(X, q)$ sending $((t_j)_j) \mapsto \sum_j t_j e_j(x)$ is well defined and bounded. Further, assume that for some choice of quotients $a_j^k(w)$, for each $k = 1, ..., J$ the series $\sum_{j=0}^{\infty}(a_j^k(w)^2)$ converges and is bounded on W. Then there exists $C > 0$ such that

$$K(w) \leq C \sum_{j=1}^{J} f_j(w)^2 \,.$$

Proof: The KL divergence is given by

$$K(w) = \frac{1}{2}\int_X (f(x,w) - f_T(x))^2 q(x)dx\,,$$

which equals

$$\begin{aligned}
\frac{1}{2}\int_X (f(x,w) - f_T(x))^2 q(x)dx &= \frac{1}{2}\int_X \left(\sum_{j=0}^{\infty} f_j(w)e_j(x)\right)^2 q(x)dx \\
&= \frac{1}{2}\left\|\tau\left(\left(f_j(w)\right)_{j=0}^{\infty}\right)\right\|^2_{L^2(X,q)} \\
&\leq C_1 \|\left(f_j(w)\right)_{j=0}^{\infty}\|^2_{\ell^2} \\
&= C_1 \sum_{j=0}^{\infty} f_j(w)^2\,,
\end{aligned}$$

for some $C_1 > 0$, since τ is bounded. Using the Cauchy-Schwartz inequality, we obtain

$$\begin{aligned}
\sum_{j=0}^{J} f_j(w)^2 + \sum_{j=J+1}^{\infty} f_j(w)^2 &= \sum_{j=0}^{J} f_j(w)^2 + \sum_{j=J+1}^{\infty}\left(\sum_{k=0}^{J} a_j^k(w) f_k(w)\right)^2 \\
&\leq \sum_{j=0}^{J} f_j(w)^2 + \sum_{j=J+1}^{\infty}\left(\sum_{k=0}^{J} a_j^k(w)^2\right)\left(\sum_{k=0}^{J} f_k(w)^2\right) \\
&= \sum_{j=0}^{J} f_j(w)^2\left(1 + \sum_{j=J+1}^{\infty}\sum_{k=0}^{J} a_j^k(w)^2\right) \\
&= \sum_{j=0}^{J} f_j(w)^2 \sum_{k=0}^{J}\sum_{j=J+1}^{\infty} a_j^k(w)^2 \\
&\leq \sup_{k=0,\ldots,J}\sup_{w\in W}\left\{\sum_{j=J+1}^{\infty} a_j^k(w)^2\right\}\left(\sum_{j=0}^{J} f_j(w)^2\right)\,,
\end{aligned}$$

where the above suprema exist by the assumption that each series $\sum_{j=0}^{\infty} a_j^k(w)^2$ is bounded on the weight space W. □

At first, the requirement that $\tau : \ell^2(\mathbb{R}) \to L^2(X,q)$ is well defined and bounded looks a little difficult. However, it is one of the easier conditions to check due to the following lemma.

Lemma 8. Suppose $\sum_{j=0}^{\infty} \|e_j(x)\|^2_{L^2(X,q)} < \infty$. Then $\tau : \ell^2(\mathbb{R}) \to L^2(X,q)$ is well defined and bounded.

Proof: If $(c_j)_j \in \ell^2(\mathbb{R})$, then for each finite n,

$$\begin{aligned}
\sum_{j=0}^{n} \|c_j e_j\|_{L^2(X,q)} &= \sum_{j=0}^{n} |c_j| \|e_j\|_{L^2(X,q)} \\
&\leq \left(\sum_{j=0}^{n} |c_j|^2\right)^{\frac{1}{2}}\left(\sum_{j=0}^{n} \|e_j\|^2_{L^2(X,q)}\right)^{\frac{1}{2}} \qquad \text{(Cauchy-Schwarz)} \\
&\leq \|(c_j)_j\|_{\ell^2}\left(\sum_{j=0}^{\infty} \|e_j\|^2_{L^2(X,q)}\right)^{\frac{1}{2}}\,.
\end{aligned}$$

This implies that the series $\sum_{j=0}^{\infty} \|c_j e_j\|_{L^2(X,q)}$ converges in $\mathbb{R}$ and so $\tau((c_j)_j) = \lim_{n\to\infty} \sum_{j=0}^{n} c_j e_j$ converges in the Hilbert space $L^2(X,q)$ and τ is well defined. Further,

$$\|\tau((c_j)_j)\|_{L^2(X,q)} \leq \sum_{j=0}^{\infty} \|c_j e_j\|_{L^2(X,q)} \leq \|(c_j)\|_{\ell^2(\mathbb{R})} \left(\sum_{j=0}^{\infty} \|e_j\|^2_{L^2(X,q)} \right)^{\frac{1}{2}},$$

so $\|\tau\|$ is bounded by $\left(\sum_{j=0}^{\infty} \|e_j\|^2_{L^2(X,q)}\right)^{\frac{1}{2}}$. □

The following lemma summarises the above results, combining them into a list of four conditions which together prove a polynomial upper bound for a model's KL divergence.

Lemma 9 (Four main conditions)**.** Suppose a model and true distribution are given by regression functions $f(x,w)$ and $f_T(x)$. If the four following conditions are satisfied

- (C1) $f(x,w) - f_T(x) = \sum_{j=0}^{\infty} f_j(w) e_j(x)$ for all $x \in X$ and $w \in W$, where each $f_j(w)$ is a polynomial,
- (C2) $\sum_{j=0}^{\infty} \|e_j(x)\|^2_{L^2(X,q)} < \infty$,
- (C3) $(f_j(w))_{j=0}^{\infty} \in \ell^2(\mathbb{R})$ for each $w \in W$,
- (C4) There exists $J > 0$ such that $\{f_1, ...f_J\}$ generates the ideal I, and a set of quotients $a_j^k(w) \in \mathbb{R}[w]$, so that for each $j \geq 0$ and k between 0 and J, $\sum_{j=0}^{\infty}(a_j^k(w)^2)$ converges and is bounded on W,

then $K(w) \leq C \sum_{j=0}^{J} f_j(w)^2$ as functions on W.

Conditions (C1)-(C3) are often not too difficult to check. We already know (C1) holds for biasless tanh networks, and we can check (C2) and (C3) using the ratio test. However, condition (C4) can be significantly harder to verify.

First, it may be hard to find a finite basis $\{f_0, ..., f_J\}$ for the ideal $I = \langle \{f_j\}_{j=0}^{\infty} \rangle$. The Hilbert basis theorem only guarantees that some finite basis exists, but the proof is not constructive [CLO15, Theorem 4]. To see if condition (C4) holds, we need to know what this basis actually is. Currently our only way to deal with this problem is to hope that the ideal I is simple enough for us to notice a pattern which helps find a generating set. This is the approach we will use when analysing two layer tanh networks in Chapter 4.

Second, even if you manage to find a finite generating set, $\{f_0, ..., f_J\}$, computing the quotients $a_j^k(w) \in \mathbb{R}[w]$ for each $k = 0, ..., 1$ and $j \geq 0$ may be difficult. The obvious thing to try is to divide each Taylor series polynomial f_j by the basis $(f_0, ..., f_J)$ using the polynomial division algorithm. However this method may give quotients which don't recreate f_j. Even if a polynomial p lies in the ideal $\langle f_0, ..., f_j \rangle$, so that $p = a_1 f_1 + ... + a_J f_J$ for some $a_1, ..., a_J \in \mathbb{R}[w]$, the division algorithm may instead output quotients $q_1, ..., q_J$ and remainder r such that

$$p = q_1 f_1 + ... q_J f_J + r\,,$$

where the remainder $r \neq 0$. If this occurs, we can't use the quotients to check condition (C4).

Example 1. Consider the polynomials $f_1 = xy + z$, $f_2 = x + y$, and $p = -y^2 + z$ in $\mathbb{R}[x, y, z]$. Then $p = f_1 - y f_2$, however when using the polynomial division algorithm (with lexicographic order), the outputs are $q_1 = q_2 = 0$ and $r = p$.

In the following Section we introduce some techniques from computational algebraic geometry to deal with this problem, culminating in Lemma 11.

3.2 Monomial orders and Gröbner bases

Despite the issues described in Section 3.1, the polynomial division algorithm still seems like our best hope for finding quotients to check condition (C4) of Lemma 9 and there are techniques which fix the problem of obtaining non-zero remainders. Specifically, Gröbner bases ensure that the division algorithm will output quotients with no remainder. In this section we define Gröbner bases, first introducing the required background on monomial orders.

A monomial order is a way of deciding which monomials are "bigger" than others, and once a monomial order is chosen, the division algorithm runs by cancelling out the largest monomial in the polynomial you are dividing, before continuing on to the smallest.

Definition 14. A *monomial order* for the polynomial ring $\mathbb{R}[x_1, ..., x_n]$ is a total ordering, $>$, on $\mathbb{Z}^n_{\geq 0}$ satisfying

- for all $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$, $\quad \alpha > \beta$ and $\gamma \in \mathbb{Z}^n_{\geq 0} \implies \alpha + \gamma > \beta + \gamma$,
- $>$ is a well ordering, meaning every subset of $\mathbb{Z}^n_{\geq 0}$ contains a smallest element.

An ordering on $\mathbb{Z}^n_{\geq 0}$ is associated to an ordering on the set of monomials in $\mathbb{R}[x_1, ...x_n]$ by identifying α with x^α.

Definition 15. In this report we will use *graded lexicographic order* (grlex). Given a multi-index $\alpha = (\alpha_1, ..., \alpha_n) \in \mathbb{Z}^n_{\geq 0}$, define $|\alpha| := \alpha_1 + ... + \alpha_n$. In graded lexicographic order, $\alpha > \beta$ if $|\alpha| > |\beta|$ or $|\alpha| = |\beta|$ and the first non-zero component of $\alpha - \beta$ is positive.

For other examples of monomial orders, and a proof that graded lexicographic order is a monomial order, see Chapter 2 of [CLO15].

An important property of graded lexicographic order is that given an index α, there are only finitely many indices β which are smaller than α. For the following definitions, we assume some monomial order has been chosen, and later in the thesis we will exclusively use graded lexicographic order.

Definition 16. Given a polynomial $p(x_1, ..., x_n) \in \mathbb{R}[x_1, ..., x_n]$ and multi-index $\alpha \in \mathbb{Z}^n_{\geq 0}$, write $p_\alpha \in \mathbb{R}$ for the coefficient of the monomial x^α in p. The polynomial p can be written as

$$p(x_1, ..., x_n) = \sum_{\alpha \in \mathbb{Z}^n_{\geq 0}} p_\alpha x^\alpha \,.$$

By definition only finitely many of the p_α's will be non-zero. The *multidegree of* p is the multi-index $\text{multideg}(p) = \max\{\alpha | \, p_\alpha \neq 0\}$, where the maximum is taken with respect to the chosen monomial order. Suppose $\alpha = \text{multideg}(p)$, then the *leading monomial of* p, $\text{LM}(p)$, is x^α, and the *leading term of* p is $\text{LT}(p) = p_\alpha x^\alpha$. Less formally, the leading term is the largest term in p according to the chosen monomial order.

Example 2. Let $p(x, y, z) = 3xy^2z - 5x^2z^2 + 2xy$. Using grlex order, $\text{multideg}(p) = (2, 0, 2)$, $\text{LT}(p) = -5x^2z^2$, and $\text{LM}(p) = x^2z^2$.

Definition 17. Given a polynomial ideal $I \subset \mathbb{R}[x_1, ..., x_n]$, $\text{LT}(I)$ is the set of all leading terms of elements of I

$$\text{LT}(I) = \{\text{LT}(p) \, | \, p \in I \backslash \{0\}\} \,,$$

and $\langle \text{LT}(I) \rangle$ is the ideal generated by this set. This is called the *leading term ideal*.

Remark. Counterintuitively, if $I = \langle f_1, ..., f_J \rangle$, it's possible that

$$\langle \text{LT}(I) \rangle \neq \langle \text{LT}(f_1), ..., \text{LT}(f_j) \rangle \,.$$

This is the case in Example 1, where $\text{LT}(p) = -y^2$ which is not an element of $\langle \text{LT}(f_1), \text{LT}(f_2) \rangle = \langle xy, x \rangle$, even though $p \in \langle f_1, f_2 \rangle$.

This causes the problem of non-zero remainders from the division algorithm. When the algorithm comes across a term which isn't divisible by a leading term of any divisors, it throws that term into the remainder.

Gröbner bases fix this problem, and make the division algorithm work as desired.

Definition 18. Given an ideal $I \subset \mathbb{R}[x_1, ..., x_n]$, a finite subset $\{g_1, ..., g_m\} \subset I$ is a *Gröbner basis* for I if

$$I = \langle g_1, ..., g_m \rangle \quad \text{and}$$
$$\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_m) \rangle .$$

The following properties of Gröbner bases are proven in Chapter 2 of [CLO15].

- Every ideal $I \subset \mathbb{R}[x_1, ..., x_n]$ has a Gröbner basis. The Hilbert basis theorem guarantees a finite basis exists, and Buchberger's algorithm converts a finite basis into a Gröbner basis by adding extra elements.
- Given a Gröbner basis $G = \{g_1, ..., g_m\}$ of I and $p \in \mathbb{R}[x_1, ..., x_n]$, there exists unique $r \in \mathbb{R}[x_1, ..., x_n]$ such that
  1. $\mathrm{LT}(g_1), ..., \mathrm{LT}(g_m)$ do not divide any of the terms of r, and
  2. There exists $g \in I$ such that $p = g + r$.

  This implies the remainder of p on division by G is unique.
- $p \in I$ if and only if the remainder on division by G is zero.

The third property solves the *ideal membership problem*, namely given any polynomial p and ideal I, you can tell if p is an element of I by dividing it by a Gröbner basis for I. If the remainder is zero, then p is an element of I, while if the remainder is non-zero, p is not an element of I.

Returning to the problem of finding the quotients for condition (C4), suppose that $\{f_0, ..., f_J\}$ is a Gröbner basis for the ideal $\langle \{f_j\}_{j=0}^{\infty} \rangle$. Then the quotients produced by the division algorithm $a_j^k(w)$ will always satisfy the following equation

$$f_j(w) = \sum_{k=0}^{J} a_j^k(w) f_k(w) ,$$

which is exactly what we need for checking (C4).

While there is no reason to assume that $\{f_0, ..., f_J\}$ is a Gröbner basis, it is easy to convert it into one using *Buchberger's algorithm* [CLO15, §2.7]. The following lemma ensures that doing so won't change the RLCT λ.

Lemma 10. Suppose $\{f_1, ..., f_t\}$ and $\{g_1, ..., g_s\}$ generate the same ideal in $\mathbb{R}[w]$. Assuming the weight space W is compact, there exists $C_1, C_2 > 0$ such that

$$C_1 \sum_{j=1}^{t} f_j(w)^2 \leq \sum_{l=1}^{s} g_l(w)^2 \leq C_2 \sum_{j=1}^{t} f_j(w)^2 ,$$

for all $w \in W$. In particular, the polynomials $H_1(w) = \sum_{j=1}^{t} f_j(w)^2$ and $H_2(w) = \sum_{l=1}^{s} g_l(w)^2$ are equivalent and have the same RLCT.

This is proven by Lin in [Lin11, Proposition 4.3].

We can use Lemma 10 to help us find a polynomial upper bound as follows. First, we convert our Hilbert basis $\{f_0, ..., f_J\}$ to a Gröbner basis $\{g_0, ..., g_s\}$. We can then divide each Taylor series polynomial by the Gröbner basis to get a set of quotients. If condition (C4) holds for these quotients, we obtain an upper bound $K(w) \leq C_1 \sum_{j=0}^{s} g_j(w)^2$. The lemma turns this into an upper bound for the original Taylor series polynomials $K(w) \leq C_2 \sum_{j=0}^{J} f_j(w)^2$.

3.3 The generic division algorithm

In this section we will assume $G = \{f_0, ..., f_J\}$ is a Gröbner basis for the ideal $\langle \{f_j\}_{j=0}^{\infty} \rangle$. We could immediately divide each polynomial f_j by this basis and find a collection of quotients $a_k^j(w)$ to check condition (C4).

One problem with jumping straight into the calculation in this way is that the quotients aren't unique. For example, we could obtain different quotients by changing the order of the divisors $(f_0, ..., f_J)$ in the division algorithm. At each step, the polynomial division algorithm takes a polynomial called the *intermediate dividend*, p, and picks the first divisor f_j whose leading term divides $\mathrm{LT}(p)$. It then subtracts from p the polynomial $\frac{\mathrm{LT}(p)}{\mathrm{LT}(f_j)} f_j$, and adds $\frac{\mathrm{LT}(p)}{\mathrm{LT}(f_j)}$ to the j^{th} quotient. Changing the order of the divisors potentially changes the choice the algorithm makes at each step, in turn changing the outputs.

Example 3. Consider the polynomials $f_1 = x^2 + yx$, $f_2 = y^2 + yx$ and $p = x^2y + yx^2$. Dividing p by (f_1, f_2) with grlex order yields $p = yf_1$, with quotients $a_1 = y$ and $a_2 = 0$. Meanwhile division of p by (f_2, f_1) gives $p = xf_2$ and quotients $a_1 = 0$ and $a_2 = x$.

Condition (C4) requires us to show that the series of quotients $\sum_{j=0}^{\infty} a_j^k(w)^2$ converges, but it is unclear whether the specific choice of quotients could affect this convergence. Perhaps the convergence could depend on the chosen order of the divisors $(f_1, ..., f_J)$ in the division algorithm. This makes it hard to know if any any observed convergence is a result of chance.

To deal with this problem we investigated modified versions of the division algorithm which work independently of the order of the divisors. The algorithm we studied is the *generic division algorithm*, which runs very similarly to the standard polynomial division algorithm, but instead of picking a single divisor f_j at each step, it averages over all divisors which "work". This algorithm was communicated to me by my supervisor Daniel Murfet.

The generic division algorithm requires us to make a few choices. First, we need to choose a monomial order with the property that for each multi-index α, there are only finitely many smaller multi-indices. As we've mentioned, graded lexicographic order has this property, and we will use this order for the rest of the Chapter.

Second, we need to choose a finite downwards closed set of multi-indices, $\Lambda \subset \mathbb{Z}^n_{\geq 0}$. Being downwards closed means that if $\alpha \in \Lambda$, every monomial less than α also lies in Λ. The previously mentioned finiteness condition on the monomial order is needed for such sets to exist in general. As a counter example, lexicographic order on $\mathbb{R}[x, y]$ doesn't have this property and there are infinitely many smaller monomials than x (specifically $y^j < x$, $\forall j > 0$). This motivates our decision to use graded lexicographic order.

Assuming we have chosen one of these sets Λ, we can then write it as

$$\Lambda = \{\alpha_1 > \alpha_2 > ... > \alpha_q\}.$$

We then choose a set of polynomials $G = \{f_1, ..., f_s\}$ in $\mathbb{R}[x_1, ..., x_n]$, to be the divisors. For each index $\beta \in \Lambda$, define the set

$$D_\beta = \left\{1 \leq j \leq s \mid \mathrm{LT}(f_j) | x^\beta\right\}.$$

This records which of the f_j's leading terms divide the monomial x^β.

Given a polynomial f for which $\mathrm{LM}(f) \in \Lambda$ we can divide f by $\{f_1, ..., f_s\}$ using the generic division algorithm as follows.

Algorithm 1 The generic division algorithm

Input: divisors $f_1, ..., f_s$, polynomial f, and $\Lambda \subset \mathbb{Z}^n_{\geq 0}$ with Λ downwards closed, finite, and $\mathrm{LM}(f) \in \Lambda$.
Output: quotients a_1, ..., a_s, remainder r

$a_1 := 0$, ..., $a_s := 0$
$r := 0$
$d_{\alpha_1} := |D_{\alpha_1}|$, ..., $d_{\alpha_q} := |D_{\alpha_q}|$
$p := f$
$i := 1$
while $i \leq q$ **do**
  if $d_{\alpha_i} \neq 0$ **then**
    $p := p - \sum_{j \in D_{\alpha_i}} \frac{1}{d_{\alpha_i}} \frac{p_{\alpha_i} x^{\alpha_i}}{\mathrm{LT}(f_j)} f_j$
    for each $j \in D_{\alpha_i}$ **do**
      $a_j := a_j + \frac{1}{d_{\alpha_i}} \frac{p_{\alpha_i} x^{\alpha_i}}{\mathrm{LT}(f_j)}$
    end for
  end if
  $i := i + 1$
end while
Output: $r := p$ and a_1, ... a_s

Definition 19. In the above algorithm, call the polynomial p the *intermediate remainder*

At the i^{th} step, the algorithm uses every divisor whose leading term divides x^{α_i}, subtracting the following sum from the intermediate remainder p

$$\frac{p_{\alpha_i}}{d_{\alpha_i}} \sum_{j \in D_{\alpha_i}} \frac{x^{\alpha_i}}{\mathrm{LT}(f_j)} f_j \,.$$

The intermediate remainder at each step is independent of any choice of order of the divisors since this sum is unaffected by reordering them. Additionally, if we do reorder the divisors, the quotients a_j are also reordered, but aren't changed in any other way.

For the generic division algorithm to be useful, it should share some of the standard polynomial division algorithm's main properties. The following proposition states firstly that the generic division algorithm outputs quotients and remainder which recreate the starting function, and secondly, the outputs have the same properties as those from the standard polynomial division algorithm.

Proposition 2. If $\mathrm{LM}(f) \in \Lambda$, then the quotients and remainder from the generic division algorithm satisfy

$$f = a_1 f_1 + ... + a_s f_s + r \,,$$

where either $r = 0$ or it is a linear combination of monomials, none of which are divisible by any of $\mathrm{LT}(f_1)$, ..., $\mathrm{LT}(f_s)$. Further, if $a_j f_j \neq 0$ then

$$\mathrm{multideg}(f) \geq \mathrm{multideg}(a_j f_j) \,.$$

Proof: Write p_i for the value of the intermediate remainder p at the i^{th} step of the algorithm, starting with $p_0 = f$. Additionally write $(a_j)_i$ for the value of the j^{th} quotient at the i^{th} step, beginning with $(a_j)_0 = 0$. At the start of the algorithm,

$$f = (a_1)_0 f_1 + ... + (a_s)_0 f_s + p_0 \,,$$

because $p_0 = f$ and $(a_1)_0 = ... = (a_s)_0 = 0$. Suppose at the i^{th} step that $f = (a_1)_i f_1 + ... + (a_s)_i f_i + p_i$. Then at the next step

$$(a_1)_{i+1} f_1 + ... + (a_s)_{i+1} f_s + p_{i+1} = \sum_{j=1}^{s} (a_j)_i f_j + \sum_{j \in D_{\alpha_{i+1}}} \frac{1}{d_{\alpha_{i+1}}} \frac{(p_i)_{\alpha_{i+1}} x^{\alpha_{i+1}}}{\mathrm{LT}(f_j)} f_j$$

$$
\begin{aligned}
&\quad + p_i - \sum_{j \in D_{\alpha_{i+1}}} \frac{1}{d_{\alpha_{i+1}}} \frac{(p_i)_{\alpha_{i+1}} x^{\alpha_{i+1}}}{\mathrm{LT}(f_j)} f_j \\
&= (a_1)_i f_1 + ... + (a_s)_i f_s + p_i \\
&= f \,.
\end{aligned}
$$

Since the statement is true for every step of the algorithm, it holds for the output, and

$$f = a_1 f_1 + ... + a_s f_s + r \,,$$

where $a_1, ..., a_s$ and r are the final quotients and remainder.

The second part of the proposition states that each term in the remainder r isn't divisible by any of the leading terms $\mathrm{LT}(f_1), ..., \mathrm{LT}(f_s)$. We will prove this through the following claim. The claim uses the chosen downwards closed set $\Lambda = \{\alpha_1 > ... > \alpha_q\}$.

Claim: For $i \geq 1$, p_i does not contain any monomials x^{α_z} for which $z \leq i$ and $D_{\alpha_z} \neq \emptyset$. Additionally $\mathrm{LT}(p_i) \leq \mathrm{LT}(f)$ for each i.

Proof of claim: We prove the claim by induction.

Base case: $i = 1$. For this base case, the first intermediate remainder is

$$p_1 = f - \sum_{j \in D_{\alpha_1}} \frac{1}{d_{\alpha_1}} \frac{f_{\alpha_1} x^{\alpha_1}}{\mathrm{LT}(f_j)} f_j \,.$$

There are three cases for the first step of the algorithm. In the first case, $x^{\alpha_1} = \mathrm{LM}(f)$ and $D_{\alpha_1} \neq \emptyset$. In this case we obtain p_1 by subtracting the leading term from f and adding on several other terms. All of these other terms are smaller as monomials than x^{α_1}, so $\mathrm{LT}(p_1) < \mathrm{LT}(f)$. This means that p_1 does not contain the monomial x^{α_1}, so p_1 satisfies the claim.

The second case is where $\mathrm{LM}(f) = x^{\alpha_1}$ but $D_{\alpha_1} = \emptyset$. The algorithm does nothing in the first step in this case, and $p_1 = f$. Clearly, p_1 contains no monomials x^{α_z} where $z \leq 1$ and $D_{\alpha_z} \neq \emptyset$. Additionally, $\mathrm{LT}(p_1) = \mathrm{LT}(f)$ because $p_1 = f$.

In the final case, $\mathrm{LM}(f) < x^{\alpha_1}$ so $f_{\alpha_1} = 0$. Again the algorithm does nothing in its first step, so $\mathrm{LT}(p_1) = \mathrm{LT}(f)$. Also, since x^{α_1} is not in p_1, the polynomial p_1 automatically does not contain any monomials x^{α_z} where $z \leq 1$ and $D_{\alpha_z} \neq \emptyset$.

There is one further case, where $\mathrm{LM}(f) > x^{\alpha_1}$, however this case can't happen due to the assumption that $\mathrm{LM}(f) \in \Lambda$.

Induction case: Assume p_i does not contain x^{α_z} for any $z \leq i$ with $D_z \neq \emptyset$ and $\mathrm{LT}(p_i) \leq \mathrm{LT}(f)$. The proof of the claim for this case is essentially the same as for the base case.

From the algorithm

$$p_{i+1} = p_i - \sum_{j \in D_{\alpha_{i+1}}} \frac{1}{d_{\alpha_{i+1}}} \frac{p_{i_{\alpha_{i+1}}} x^{\alpha_{i+1}}}{\mathrm{LT}(f_j)} f_j \,.$$

There are again three cases. In the first case, $p_{i_{\alpha_{i+1}}} \neq 0$ and $D_{\alpha_{i+1}} \neq \emptyset$. Here, the algorithm cancels out the $x^{\alpha_{i+1}}$ term from p_i, so that p_{i+1} does not contain $x^{\alpha_{i+1}}$ as a monomial. Since p_i doesn't contain any x^{α_z} where $z \leq i+1$ and $D_z \neq \emptyset$, it follows that p_{i+1} doesn't contain any x^{α_z} where $z \leq i$ and $D_{\alpha_z} \neq \emptyset$. Additionally, each term added to p_i in this step is smaller than $\mathrm{LT}(p_i)$ which itself is smaller than $\mathrm{LT}(f)$. Hence $\mathrm{LT}(p_{i+1}) \leq \mathrm{LT}(f)$.

In the second case, $D_{\alpha_{i+1}} = \emptyset$, and so $p_{i+1} = p_i$ and the intermediate polynomial p_{i+1} automatically satisfies the claim.

Finally, the third case is where $p_{i_{\alpha_{i+1}}} = 0$, implying $p_{i+1_{\alpha_{i+1}}} = 0$, and the algorithm does nothing at this step so $p_{i+1} = p_i$. Again, p_{i+1} will satisfy both parts of the claim.

We can conclude that at any step of the algorithm, p_i doesn't contain any monomials x^{α_z} where $z \leq i$ and $D_{\alpha_z} \neq \emptyset$, and additionally $\mathrm{LT}(p_i) \leq \mathrm{LT}(f)$. □

The above claim tells us that $r = p_q$ contains no monomials of the form x^{α_z} for which $z \leq q$ and $D_{\alpha_z} \neq \emptyset$. Additionally, r contains no monomials larger than x^{α_1} since it is constructed by adding terms to f which are less than or equal to x^{α_1}. Hence each term of r is not divisible by any of $\mathrm{LT}(f_1), ..., \mathrm{LT}(f_s)$.

For the last part of the proposition that $\mathrm{multideg}(a_j f_j) \leq \mathrm{multideg}(f)$, note that a_j is given by a linear combination of monomials of the form

$$\frac{x^{\alpha_i}}{\mathrm{LT}(f_j)},$$

and the coefficient of the above term in the quotient a_j can only be non-zero when x^{α_i} appears in one of the p_l's. Since $\mathrm{LT}(p_l) \leq \mathrm{LT}(f)$ for all $1 \leq l \leq q$, the quotient a_j will be made up of monomials less than or equal to $\frac{\mathrm{LT}(f)}{\mathrm{LT}(f_j)}$. Hence $\mathrm{multideg}(a_j) \leq \mathrm{multideg}(f)$. □

3.3.1 Steps in the algorithm

We can unravel the generic division algorithm to find expressions for the intermediate remainder p_i at each step. This will help us find expressions for the quotients output by the algorithm, and we will be able to use these expressions to check the convergence of the series in condition (C4).

For this section, we fix a set of divisors, $\{f_1, ..., f_s\} \subset \mathbb{R}[x_1, ..., x_n]$ and a downwards closed set of multi-indices $\Lambda = \{\alpha_1 > ... > \alpha_q\}$.

Definition 20. For each α and β in $\mathbb{Z}^n_{\geq 0}$ define

$$\tau_{\alpha,\beta,j} = \left(\frac{x^\alpha}{\mathrm{LT}(f_j)} f_j\right)_\beta,$$

where $1 \leq j \leq s$, and $j \in D_\alpha$. The notation $(-)_\beta$ refers to the coefficient of x^β in the polynomial that lies inside the parentheses. We then define

$$\tau_{\alpha,\beta} = \sum_{j \in D_\alpha} \tau_{\alpha,\beta,j},$$

and set $\tau_{\alpha,\beta} = 0$ if $D_\alpha = \emptyset$.

We will use these $\tau_{\alpha,\beta}$ factors to first find expressions for p_1, p_2 and p_3. Using these three examples, we can search for a general pattern and use it to find an expression for an arbitrary p_i.

Starting with the first intermediate remainder, we have

$$p_1 = f - \sum_{j \in D_{\alpha_1}} \frac{1}{d_{\alpha_1}} \frac{f_{\alpha_1} x^{\alpha_1}}{\mathrm{LT}(f_j)} f_j$$

$$\begin{aligned}(p_1)_{\alpha_2} &= f_{\alpha_2} - \sum_{j \in D_{\alpha_1}} f_{\alpha_1} \frac{1}{d_{\alpha_1}} \left(\frac{x^{\alpha_1}}{\mathrm{LT}(f_j)} f_j\right)_{\alpha_2} \\ &= f_{\alpha_2} - \frac{f_{\alpha_1}}{d_{\alpha_1}} \sum_{j \in D_{\alpha_1}} \tau_{\alpha_1,\alpha_2,j}\end{aligned}$$

$$= f_{\alpha_2} - \frac{f_{\alpha_1}}{d_{\alpha_1}} \tau_{\alpha_1,\alpha_2} \, .$$

We can then write p_2 as

$$\begin{aligned}
p_2 &= p_1 - \sum_{j \in D_{\alpha_2}} \frac{1}{d_{\alpha_2}} \frac{(p_1)_{\alpha_2}}{\mathrm{LT}(f_j)} f_j \\
&= f - \sum_{j \in D_{\alpha_1}} \frac{1}{d_{\alpha_1}} \frac{f_{\alpha_1} x^{\alpha_1}}{\mathrm{LT}(f_j)} f_j - \sum_{j \in D_{\alpha_2}} \frac{1}{d_{\alpha_2}} \frac{(p_1)_{\alpha_2} x^{\alpha_2}}{\mathrm{LT}(f_j)} f_j \\
&= f - \sum_{j \in D_{\alpha_1}} \frac{1}{d_{\alpha_1}} \frac{f_{\alpha_1} x^{\alpha_1}}{\mathrm{LT}(f_j)} f_j - \sum_{j \in D_{\alpha_2}} \frac{1}{d_{\alpha_2}} \left(f_{\alpha_2} - \frac{f_{\alpha_1}}{d_{\alpha_1}} \tau_{\alpha_1,\alpha_2} \right) \frac{x^{\alpha_2}}{\mathrm{LT}(f_j)} f_j \\
&= f - \sum_{j \in D_{\alpha_1}} \frac{1}{d_{\alpha_1}} \frac{f_{\alpha_1} x^{\alpha_1}}{\mathrm{LT}(f_j)} f_j - \sum_{j \in D_{\alpha_2}} \frac{f_{\alpha_2}}{d_{\alpha_2}} \frac{x^{\alpha_2}}{\mathrm{LT}(f_j)} f_j + \sum_{j \in D_{\alpha_2}} \frac{f_{\alpha_1}}{d_{\alpha_1} d_{\alpha_2}} \tau_{\alpha_1,\alpha_2} \frac{x^{\alpha_2}}{\mathrm{LT}(f_j)} f_j \, .
\end{aligned}$$

This gives

$$\begin{aligned}
(p_2)_{\alpha_3} &= f_{\alpha_3} - \sum_{j \in D_{\alpha_1}} \frac{f_{\alpha_1}}{d_{\alpha_1}} \tau_{\alpha_1,\alpha_3,j} - \sum_{j \in D_{\alpha_2}} \frac{f_{\alpha_2}}{d_{\alpha_2}} \tau_{\alpha_2,\alpha_3,j} + \sum_{j \in D_{\alpha_2}} \frac{f_{\alpha_1}}{d_{\alpha_1} d_{\alpha_2}} \tau_{\alpha_1,\alpha_2} \tau_{\alpha_2,\alpha_3,j} \\
&= f_{\alpha_3} - \frac{1}{d_{\alpha_1}} f_{\alpha_1} \tau_{\alpha_1,\alpha_3} - \frac{1}{d_{\alpha_2}} f_{\alpha_2} \tau_{\alpha_2,\alpha_3} + \frac{1}{d_{\alpha_1} d_{\alpha_2}} f_{\alpha_1} \tau_{\alpha_1,\alpha_2} \tau_{\alpha_2,\alpha_3} \, ,
\end{aligned}$$

and then

$$\begin{aligned}
p_3 &= p_2 - \sum_{j \in D_{\alpha_3}} \frac{1}{d_{\alpha_3}} \frac{(p_2)_{\alpha_3} x^{\alpha_3}}{\mathrm{LT}(f_j)} f_j \\
&= f - \sum_{j \in D_{\alpha_1}} \frac{1}{d_{\alpha_1}} \frac{f_{\alpha_1} x^{\alpha_1}}{\mathrm{LT}(f_j)} f_j - \sum_{j \in D_{\alpha_2}} \left(\frac{1}{d_{\alpha_2}} f_{\alpha_2} - \frac{1}{d_{\alpha_1} d_{\alpha_2}} \tau_{\alpha_1,\alpha_2} f_{\alpha_1} \right) \frac{x^{\alpha_2}}{\mathrm{LT}(f_j)} f_j \\
&\quad - \sum_{j \in D_{\alpha_3}} \frac{1}{d_{\alpha_3}} \left(f_{\alpha_3} - \frac{1}{d_{\alpha_1}} f_{\alpha_1} \tau_{\alpha_1,\alpha_3} - \frac{1}{d_{\alpha_2}} f_{\alpha_2} \tau_{\alpha_2,\alpha_3} + \frac{1}{d_{\alpha_1} d_{\alpha_2}} f_{\alpha_1} \tau_{\alpha_1,\alpha_2} \tau_{\alpha_2,\alpha_3} \right) \frac{x^{\alpha_3}}{\mathrm{LT}(f_j)} f_j \\
&= f - \sum_{t=1}^{3} \sum_{j \in D_{\alpha_t}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} f_j + \sum_{1 \le s < t \le 3} \sum_{j \in D_{\alpha_t}} \frac{1}{d_{\alpha_s} d_{\alpha_t}} f_{\alpha_s} \tau_{\alpha_s,\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} f_j \\
&\quad - \sum_{j \in D_{\alpha_3}} \frac{1}{d_{\alpha_1} d_{\alpha_2} d_{\alpha_3}} f_{\alpha_1} \tau_{\alpha_1,\alpha_2} \tau_{\alpha_2,\alpha_3} \frac{x^{\alpha_3}}{\mathrm{LT}(f_j)} f_j \, .
\end{aligned}$$

These appear to follow a pattern. The i^{th} intermediate remainder p_i looks like it equals f with i different sums added or subtracted to it. The first of these is a sum over each integer t lying between 1 and i, while the second is a sum over all pairs of distinct integers between 1 and i. The next is a sum over all triples of distinct integers between 1 and i. This continues until we reach a sum over all i-tuples of distinct integers between 1 and i. To describe these sums precisely we use the following definitions.

Definition 21. A *chain of length m* is a descending chain of m multi-indices in $\mathbb{Z}^n_{\geq 0}$, written $\beta_1 > \beta_2 > ... \beta_m$. If the endpoints of a chain satisfy the constraints $\gamma \geq \beta_1$ and $\beta_m \geq \alpha$, we write the chain as $\gamma \geq \beta_1 > ... > \beta_m \geq \alpha$ and describe it as a *chain of length m between γ and α*.

Definition 22. We will frequently come across sums of the following form

$$\sum_{\alpha_1 \geq \beta_1 > ... > \beta_m \geq \alpha_i} \sum_{j \in D_{\beta_m}} \frac{(-1)^m}{d_{\beta_1} ... d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2} ... \tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j \, .$$

We will refer to this as the *sum over all chains of length m between α_1 and α_q*.

Proposition 3. In the generic division algorithm, the intermediate remainder at the i^{th} step is given by

$$\begin{aligned} p_i =& f - \sum_{1\le t\le i} \sum_{j\in D_{\alpha_t}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} f_j \\ &+ \sum_{m=2}^{i} \sum_{\alpha_1\ge\beta_1>\ldots>\beta_m\ge\alpha_i} \sum_{j\in D_{\beta_m}} \frac{(-1)^m}{d_{\beta_1}\ldots d_{\beta_m}} f_{\beta_1}\tau_{\beta_1,\beta_2}\ldots\tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j \,. \end{aligned} \tag{3.3.1}$$

The second sum in the above expression adds over all chains of length 2 to i with endpoints between α_1 and α_i.

Proof: By induction. We have just seen that the claim holds for p_1, p_2, and p_3, which we use as the base case.

For the induction case, suppose that the expression (3.3.1) holds for p_i. Then from the algorithm

$$p_{i+1} = p_i - \sum_{j\in D_{\alpha_{i+1}}} \frac{1}{d_{\alpha_{i+1}}} \frac{(p_i)_{\alpha_{i+1}}}{\mathrm{LT}(f_j)} f_j \tag{3.3.2}$$

and using (3.3.1),

$$\begin{aligned} (p_i)_{\alpha_{i+1}} =& f_{\alpha_{i+1}} - \sum_{1\le t\le i} \sum_{j\in D_{\alpha_t}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \left(\frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} f_j\right)_{\alpha_{i+1}} \\ &+ \sum_{m=2}^{i} \sum_{\alpha_1\ge\beta_1>\ldots>\beta_m\ge\alpha_i} \sum_{j\in D_{\beta_m}} \frac{(-1)^m}{d_{\beta_1}\ldots d_{\beta_m}} f_{\beta_1}\tau_{\beta_1,\beta_2}\ldots\tau_{\beta_{m-1},\beta_m} \left(\frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j\right)_{\alpha_{i+1}} \\ =& f_{\alpha_{i+1}} - \sum_{1\le t\le i} \frac{1}{d_{\alpha_t}} f_{\alpha_t}\tau_{\alpha_t,\alpha_{i+1}} + \sum_{m=2}^{i} \sum_{\alpha_1\ge\beta_1>\ldots>\beta_m\ge\alpha_i} \frac{(-1)^m}{d_{\beta_1}\ldots d_{\beta_m}} f_{\beta_1}\tau_{\beta_1,\beta_2}\ldots\tau_{\beta_{m-1},\beta_m}\tau_{\beta_m,\alpha_{i+1}} \,. \end{aligned}$$

Substituting this and (3.3.1) into (3.3.2) gives

$$\begin{aligned} p_{i+1} =& f - \sum_{1\le t\le i} \sum_{j\in D_{\alpha_t}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} f_j \\ &+ \sum_{m=2}^{i} \sum_{\alpha_1\ge\beta_1>\ldots>\beta_m\ge\alpha_i} \sum_{j\in D_{\beta_m}} \frac{(-1)^m}{d_{\beta_1}\ldots d_{\beta_m}} f_{\beta_1}\tau_{\beta_1,\beta_2}\ldots\tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j \\ &- \sum_{j\in D_{\alpha_{i+1}}} \frac{1}{d_{\alpha_{i+1}}} \Bigg(f_{\alpha_{i+1}} - \sum_{1\le t\le i} \frac{1}{d_{\alpha_t}} f_{\alpha_t}\tau_{\alpha_t,\alpha_{i+1}} + \\ &\qquad \sum_{m=2}^{i} \sum_{\alpha_1\ge\beta_1>\ldots>\beta_m\ge\alpha_i} \frac{(-1)^m}{d_{\beta_1}\ldots d_{\beta_m}} f_{\beta_1}\tau_{\beta_1,\beta_2}\ldots\tau_{\beta_{m-1},\beta_m}\tau_{\beta_m,\alpha_{i+1}} \Bigg) \frac{x^{\alpha_{i+1}}}{\mathrm{LT}(f_j)} f_j \,. \end{aligned} \tag{3.3.3}$$

We can write this as

$$\begin{aligned} p_{i+1} = f - \sum_{1\le t\le i+1} \sum_{j\in D_{\alpha_t}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} f_j \\ + (a) + (b) + (c) \,, \end{aligned}$$

where

$$(a) = \sum_{1\le t\le i} \sum_{j\in D_{\alpha_{i+1}}} \frac{1}{d_{\alpha_t} d_{\alpha_{i+1}}} \tau_{\alpha_t,\alpha_{i+1}} \frac{x^{\alpha i+1}}{\mathrm{LT}(f_j)} f_j \,,$$

$$(b) = -\sum_{m=2}^{i} \sum_{\alpha_1 \geq \beta_1 > \ldots > \beta_m \geq \alpha_i} \sum_{j \in D_{\alpha_{i+1}}} \frac{(-1)^m}{d_{\beta_1} \ldots d_{\beta_m} d_{\alpha_{i+1}}} f_{\beta_1} \tau_{\beta_1,\beta_2} \ldots \tau_{\beta_{m-1},\beta_m} \tau_{\beta_m,\alpha_{i+1}} \frac{x^{\alpha_{i+1}}}{\mathrm{LT}(f_j)} f_j \,,$$

$$(c) = \sum_{m=2}^{i} \sum_{\alpha_1 \geq \beta_1 > \ldots > \beta_m \geq \alpha_i} \sum_{j \in D_{\beta_m}} \frac{(-1)^m}{d_{\beta_1} \ldots d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2} \ldots \tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j \,.$$

Term (a) is the sum over all chains of length 2 whose end point is exactly equal to α_{i+1}. Similarly, term (b) is the sum over all chains of lengths 3 to $i+1$ whose end point equals α_{i+1}. Hence $(a)+(b)$ is the sum of all chains lengths 2 to $i+1$ with end point equal to α_{i+1}.

$$(a)+(b) = \sum_{m=2}^{i+1} \sum_{\substack{\alpha_1 \geq \beta_1 > \ldots > \beta_m \geq \alpha_{i+1} \\ \text{s.t. } \beta_m = \alpha_{i+1}}} \frac{(-1)^m}{d_{\beta_1} \ldots d_{\beta_m}} \sum_{j \in D_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2} \ldots \tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j \,.$$

Meanwhile, the third term (c) is the sum over all chains of lengths 2 to i with end point greater than or equal to α_i. There are no chains of length $i+1$ with endpoint strictly greater than α_{i+1} (a set of $i+1$ distinct indices placed in decreasing order with starting point α_1 will always have endpoint β with $\alpha_{i+1} \geq \beta$). Hence we are free to describe (c) as the sum over all chains lengths 2 to $i+1$ with endpoint strictly greater than α_{i+1}. Adding (a), (b), and (c) together gives the sum over all chains of lengths 2 to $i+1$ with endpoints between α_1 and α_{i+1}

$$(a)+(b)+(c) = \sum_{m=2}^{i+1} \sum_{\alpha_1 \geq \beta_1 > \ldots > \beta_m \geq \alpha_{i+1}} \sum_{j \in D_{\beta_m}} \frac{(-1)^m}{d_{\beta_1} \ldots d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2} \ldots \tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j \,.$$

Plugging this back into (3.3.3) gives

$$\begin{aligned} p_{i+1} = f &- \sum_{1 \leq t \leq i+1} \sum_{j \in D_{\alpha_t}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} f_j \\ &+ \sum_{m=2}^{i+1} \sum_{\alpha_1 \geq \beta_1 > \ldots > \beta_m \geq \alpha_{i+1}} \sum_{j \in D_{\beta_m}} \frac{(-1)^m}{d_{\beta_1} \ldots d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2} \ldots \tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j \,, \end{aligned}$$

proving the claim. □

Corollary. The remainder $r = p_q$ is also given by the form in (3.3.1).

Corollary. Suppose $I \subset \mathbb{R}[x_1, \ldots, x_n]$ is an ideal and $f \in I$, if $\{f_1, \ldots, f_s\}$ is a Gröbner basis for I. Then the remainder, p_q, when applying the generic division algorithm to f with this basis is zero, so

$$\begin{aligned} f = &\sum_{1 \leq t \leq q} \sum_{j \in D_{\alpha_t}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} f_j \\ &- \sum_{m=2}^{q} \sum_{\alpha_1 \geq \beta_1 > \ldots > \beta_m \geq \alpha_q} \sum_{j \in D_{\beta_m}} \frac{(-1)^m}{d_{\beta_1} \ldots d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2} \ldots \tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} f_j \,. \end{aligned}$$

In particular

$$f = \sum_{j=1}^{s} q_j f_j \,,$$

with quotients given by

$$q_j = \sum_{\substack{1 \leq t \leq q \\ \text{s.t.} j \in D_{\alpha_t}}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} - \sum_{m=2}^{q} \sum_{\substack{\alpha_1 \geq \beta_1 > \ldots > \beta_m \geq \alpha_q \\ \text{s.t.} j \in D_{\beta_m}}} \frac{(-1)^m}{d_{\beta_1} \ldots d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2} \ldots \tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)} \,. \tag{3.3.4}$$

Remark. The expression (3.3.1) does not depend on the choice of finite downwards closed set $\Lambda \subset \mathbb{Z}^m_{\geq 0}$ as long as $\mathrm{LM}(f) \in \Lambda$. If $\Lambda = \{\alpha_1 > ... > \alpha_q\}$ and Λ' is another finite downwards closed set which contains Λ as a subset, we can write it as $\Lambda' = \{\beta_1 > ... > \beta_t > \alpha_1 > ... > \alpha_q\}$. When the generic division algorithm is run with the same f and divisors $f_1, ..., f_s$, but with Λ' in place of Λ, it does nothing until it reaches the index $\alpha_i = \mathrm{LM}(f)$ before proceeding in the exact same manner.

This lets us drop the choice of downwards closed set Λ from the notation. We can rewrite the first term in (3.3.4) as

$$\sum_{\substack{1\leq t\leq q \\ \text{s.t.} j\in D_{\alpha_t}}} \frac{1}{d_{\alpha_t}} f_{\alpha_t} \frac{x^{\alpha_t}}{\mathrm{LT}(f_j)} = \sum_{\substack{\alpha \\ \text{s.t. } j\in D_\alpha}} \frac{1}{d_\alpha} f_\alpha \frac{x^\alpha}{\mathrm{LT}(f_j)},$$

where $\sum_\alpha$ is the sum over all multi-indices in $\mathbb{Z}^n_{\geq 0}$. The sum on the right is finite since $f_\alpha \neq 0$ for only finitely many indices α. While the left hand side is written in terms of a chosen $\Lambda = \{\alpha_1 > ... > \alpha_q\}$, Λ is assumed to contain every monomial in f. Hence the two sides of the equation are the same.

Likewise, the other term can be written

$$-\sum_{m=2}^{q} \sum_{\substack{\alpha_1\geq\beta_1>...>\beta_m\geq\alpha_q \\ \text{s.t.} j\in D_{\beta_m}}} \frac{(-1)^m}{d_{\beta_1}...d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2}...\tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)}$$

$$= \sum_{m=2}^{\infty} \sum_{\substack{\beta_1>...>\beta_m \\ \text{s.t.} j\in D_{\beta_m}}} \frac{(-1)^{m+1}}{d_{\beta_1}...d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2}...\tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)},$$

where $\sum_{\beta_1>...>\beta_m}$ is the sum over all chains of length m, without any restrictions on the end points.

Again, the right hand side is a finite sum because f contains only finitely many monomials, and for each monomial in f there are only finitely many smaller monomials when grlex is used. Additionally, since each monomial of f, α, is assumed to be in Λ, every chain starting at α has no more than q multi-indices and so there are no extra chains included on the right hand side. Hence the two sides are equal. The quotients can then be written as

$$q_j = \sum_{\substack{\alpha \\ \text{s.t. } j\in D_\alpha}} \frac{1}{d_\alpha} f_\alpha \frac{x^\alpha}{\mathrm{LT}(f_j)} + \sum_{m=2}^{\infty} \sum_{\substack{\beta_1>...>\beta_m \\ \text{s.t.} j\in D_{\beta_m}}} \frac{(-1)^{m+1}}{d_{\beta_1}...d_{\beta_m}} f_{\beta_1} \tau_{\beta_1,\beta_2}...\tau_{\beta_{m-1},\beta_m} \frac{x^{\beta_m}}{\mathrm{LT}(f_j)}. \tag{3.3.5}$$

3.3.2 Division graphs

A division graph is a useful way to visualise the expressions coming from the generic division algorithm, especially the "chains" of indices made up of the $\tau_{\alpha,\beta}$ terms from Definition 20.

Definition 23. Given a Gröbner basis $G = \{f_1, ..., f_s\}$ in $\mathbb{R}[x_1, ..., x_n]$, *the division graph* for G is the oriented graph which has $\mathbb{Z}^n_{\geq 0}$ as the vertices, and an arrow from index β to index α if $\tau_{\alpha,\beta} \neq 0$.

Note that $\tau_{\alpha,\beta}$ can only be non-zero if $\alpha \geq \beta$ in the chosen monomial order. Hence the division graph will only contain arrows pointing in non-decreasing directions.

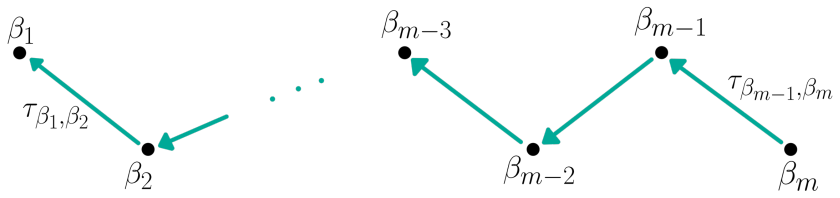


Figure 3.1: A path in the division graph associated to the chain $\beta_1 > \beta_2 > ... > \beta_m$.

Consider the sum over all chains of length m

$$\sum_{\substack{\beta_1>\ldots>\beta_m \\ \text{s.t.} j\in D_{\beta_m}}} \frac{(-1)^{m+1}}{d_{\beta_1}\ldots d_{\beta_m}} f_{\beta_1}\tau_{\beta_1,\beta_2}\ldots\tau_{\beta_{m-1},\beta_m}\frac{x^{\beta_m}}{\mathrm{LT}(f_j)}f_j\,. \tag{3.3.6}$$

If $\tau_{\beta_l,\beta_{l+1}}=0$ for any pair of adjacent indices in a chain, that chain's term in the sum will be zero. Hence (3.3.6) can be thought of as the sum over all paths in the division graph containing m distinct indices.

3.4 The generic division algorithm for polynomial upper bounds

The generic division algorithm gives expressions for the quotients (3.3.5) which we can use to check convergence condition (C4). First we introduce some definitions to simplify the notation.

Definition 24. Given two multi-indices $\alpha>\beta$ in $\mathbb{Z}^n_{\geq 0}$, define the real number

$$K(\alpha,\beta)=\sum_{m=2}^{\infty}\sum_{\substack{\beta_1>\ldots>\beta_m \\ \beta_1=\alpha,\beta_m=\beta}}\frac{(-1)^{m+1}}{d_{\beta_1}\ldots d_{\beta_m}}\tau_{\beta_1,\beta_2}\ldots\tau_{\beta_{m-1},\beta_m}\,.$$

We also define

$$K(\alpha,\alpha)=\begin{cases}\frac{1}{d_\alpha}\,, & d_\alpha\neq 0\\ 0\,, & d_\alpha=0\end{cases}\,.$$

$K(\alpha,\beta)$ can only be non-zero when there is a path from β to α in the division graph. Additionally, the value of each $K(\alpha,\beta)$ only depends on the Gröbner basis $G=\{f_1,\ldots,f_s\}$, and not the polynomial f which is being divided.

This notation lets us write the expressions for the quotients (3.3.5) as

$$q_j=\sum_{\substack{\alpha\geq\beta \\ \text{s.t. } j\in D_\beta}} K(\alpha,\beta)f_\alpha\frac{x^\beta}{\mathrm{LT}(f_j)}\,.$$

Returning to problem of finding a polynomial upper bound for the KL divergence, we will replace the variable x with w to emphasise that the polynomials are functions of the weights of a neural network. As we had before, we have a sequence of polynomials $\{f_j\}_{j=0}^{\infty}$, and we can assume that $G=\{f_0,\ldots,f_J\}$ is a Gröbner basis for their ideal $I=\langle\{f_j\}_{j=0}^{\infty}\rangle$. For each $j>J$, we use the generic division algorithm to write

$$f_j=\sum_{k=0}^{J}a_j^k f_k\,,$$

where

$$a_j^k=\sum_{\substack{\alpha\geq\beta \\ \text{s.t. } k\in D_\beta}} K(\alpha,\beta)(f_j)_\alpha\frac{w^\beta}{\mathrm{LT}(f_k)}\,.$$

To simplify the expressions, for each $\alpha\in\mathbb{Z}^n_{\geq 0}$ we define

$$\Omega_\alpha^k:=\sum_{\substack{\beta\leq\alpha \\ k\in D_\beta}} K(\alpha,\beta)\frac{w^\beta}{\mathrm{LT}(f_k)}\,, \tag{3.4.1}$$

so then

$$a_j^k=\sum_\alpha (f_j)_\alpha\Omega_\alpha^k\,.$$

To show condition (C4) we must bound the series $\sum_{j=0}^{\infty}(a_j^k)^2$ for each $k = 0, 1, ..., J$. We can rewrite this series as

$$\sum_{j=0}^{\infty}(a_j^k)^2 = \sum_{j=0}^{\infty}\left(\sum_{\alpha}(f_j)_\alpha \Omega_\alpha^k\right)^2 .$$

For each j define the set $H_j = \{\alpha \in \mathbb{Z}_{\geq 0}^n \mid (f_j)_\alpha \neq 0\}$, so that

$$\begin{aligned}\sum_{j=0}^{\infty}(a_j^k)^2 &= \sum_{j=0}^{\infty}\left(\sum_{\alpha\in H_j}(f_j)_\alpha \Omega_\alpha^k\right)^2 \\ &\leq \sum_{j=0}^{\infty}\left(\sum_{\alpha\in H_j}(f_j)_\alpha^2\right)\left(\sum_{\alpha\in H_j}\left(\Omega_\alpha^k\right)^2\right) .\end{aligned}$$

Lemma 11. Condition (C4) will hold if for each $k = 0, ..., J$, the series $\sum_{j=0}^{\infty}\left(\sum_{\alpha\in H_j}(f_j)_\alpha^2\right)\left(\sum_{\alpha\in H_j}\left(\Omega_\alpha^k\right)^2\right)$ converges and is bounded for all $w \in W$.

In the next Chapter, we will use this lemma to find polynomial upper bounds for two layer networks.

Chapter 4

Examples

In the previous Chapters, we saw that the Taylor series coefficients for biasless two layer tanh networks are polynomials of the weights, and found conditions (C1)-(C4) in Lemma 9, which could provide a polynomial upper bound for these models' KL divergence. In this Chapter we check these conditions for a variety of two layer networks.

The first model we will study is a two neuron network with sine activation functions studied in [Wat09]. We will use Gröbner bases and the generic division algorithm to find a polynomial upper bound for this model, and the results will easily carry over to a similar network with tanh neurons. To study larger tanh networks with arbitrarily many neurons though, the generic division algorithm becomes computationally unworkable, and we require a different approach.

In Sections 4.1-4.4 the polynomial upper bounds will apply when the true function $f_T(x)$ is zero everywhere. In the final part of this Chapter, Section 4.5, we will extend our results to models where the true function is non-zero, leading to the Main Theorem.

4.1 Example 1: Two neuron sine network

In this section we use the generic division algorithm to show conditions (C1)-(C4) are satisfied in [Wat09, Example 3.2].

In this example, Watanabe introduces a two layer neural network which has sine as its activation function. He uses this example to illustrate Hilbert's basis theorem, showing that the model's Taylor series splits into a sum of polynomials of the weights and functions of the input (like the form in the Replacement Strategy), and finds a finite generating set for these Taylor series polynomials. However, he does not discuss this model's KL divergence.

This is a convenient model to test our methods on. Unlike tanh networks, its Taylor series converges globally, so we don't need to worry about its domain of convergence. Additionally, although sine activation functions are quite niche, we will be able to quickly adapt the results from this model to study tanh networks with two neurons.

The model is given by the regression function

$$f(x, a, b, c, d) = a \sin(bx) + c \sin(dx)\,,$$

where $x \in [-1, 1]$ is the input and $w = (a, b, c, d) \in \mathbb{R}^4$ is the weight.

If we set the input distribution, $q(x)$ to be uniform on $[-1, 1]$, and the true regression function f_T to be zero, the KL divergence for the model is

$$K(w) = \frac{1}{2} \int_{[-1,1]} (f(x, w))^2 dx\,.$$

While we didn't discuss sine activation functions in Chapter 2, it is easy to see that the Taylor series coefficients for f are polynomials of the weights. The Taylor series for f with respect to x about $x = 0$ is

$$f(x, a, b, c) = \sum_{j=0}^{\infty} \frac{(-1)^j}{(2j+1)!} x^{2j+1} \left(ab^{2j+1} + cd^{2j+1}\right) .$$

Defining $e_j(x) = \frac{(-1)^j}{\sqrt{(2j+1)!}} x^{2j+1}$ and $f_j(a, b, c, d) = \frac{1}{\sqrt{(2j+1)!}} \left(ab^{2j+1} + cd^{2j+1}\right)$ we have that

$$f(x, w) = \sum_{j=0}^{\infty} f_j(w) e_j(x) , \tag{4.1.1}$$

so the model satisfies (C1). There is clearly some choice in how the constants $\frac{1}{(2j+1)!}$ can be split between $e_j(x)$ and $f_j(w)$, and we will examine this choice more closely for tanh networks in Section 4.2. For the meantime though, the choice we have made ensures that $\sum_{j=0}^{\infty} \|e_j\|^2_{L^2(X,q)} < \infty$, since

$$\begin{aligned} \sum_{j=0}^{\infty} \|e_j(x)\|^2_{L^2(X,q)} &= \sum_{j=0}^{\infty} \left(\int_{-1}^{1} \frac{(-1)^{2j}}{(2j+1)!} x^{4j+2} dx \right) \\ &= \sum_{j=0}^{\infty} \frac{1}{(2j+1)!} \left[\frac{2}{4j+3} \right] , \end{aligned}$$

which converges by the ratio test. Hence this model satisfies condition (C2). Meanwhile,

$$\|(f_j(w))_j\|^2_{\ell^2} = \sum_{j=0}^{\infty} \frac{1}{(2j+1)!} (ab^{2j+1} + cd^{2j+1})^2 .$$

For any fixed value of (a, b, c, d), choose a constant $M > \max\{|a|, |b|, |c|, |d|\}$, then

$$\|(f_j(w))_j\|^2_{\ell^2} \leq \sum_{j=0}^{\infty} \frac{1}{(2j+1)!} 2M^{4j+4} .$$

Using the ratio test, the series on the right converges for any value of M. Hence $(f_j(w))_{j=0}^{\infty} \in \ell^2(\mathbb{R})$ for any value of w and the model satisfies condition (C3).

4.1.1 Applying the generic division algorithm

Since the two neuron sine model satisfies conditions (C1)-(C3), we just need to prove its Taylor series polynomials $f_j(a, b, c, d)$ satisfy condition (C4) to prove a polynomial upper bound on the model's KL divergence. Our strategy is to first find a finite basis for the ideal $I = \langle \{f_j(w)\}_{j=0}^{\infty} \rangle$, convert this to a Gröbner basis, find quotients using the generic division algorithm, and finally check all the convergence conditions using these quotients.

For the first step, Watanabe has shown that $\{f_0, f_1\}$ is a basis for this ideal I [Wat09, p. 80]. Buchberger's algorithm [CLO15, §2.7] converts this to the Gröbner basis $G' = \{ab + cd, ab^3 + cd^3, b^2cd - cd^3, acd^3 + bc^2d^2\}$ when using graded lexicographic order. Noting that $\mathrm{LT}(ab^3 + cd^3) \in \langle \mathrm{LT}(G') \setminus \{ab^3 + cd^3\} \rangle$, we can simplify the Gröbner basis to

$$G = \{g_1, g_2, g_3\} = \{ab + cd, b^2cd - cd^3, acd^3 + bc^2d^2\} . \tag{4.1.2}$$

The leading terms of this basis are $\mathrm{LT}(g_1) = ab$, $\mathrm{LT}(g_2) = b^2cd$, and $\mathrm{LT}(g_3) = acd^3$.

Finding the division graph

The next step is to compute the division graph for the Gröbner basis G, specifically by finding the $\tau_{\alpha,\beta}$ values. This will let us compute the $K(\alpha,\beta)$ values and Ω^k_α polynomials.

Let $\alpha > \beta$ be any two multi-indices in $\mathbb{Z}^4_{\geq 0}$. Then $\mathrm{LT}(g_1)|w^\alpha \iff \alpha = (1,1,0,0) + \gamma$ for some $\gamma \in \mathbb{Z}^4_{\geq 0}$. Then

$$\begin{aligned}
\tau_{\alpha,\beta,1} &= \left(\frac{w^\alpha}{ab}(ab+cd)\right)_\beta \\
&= \left(w^\alpha + \frac{w^{(1,1,0,0)+\gamma}}{w^{(1,1,0,0)}}cd\right)_\beta \\
&= (w^\gamma cd)_\beta \\
&= \begin{cases} 1\,, & \gamma + (0,0,1,1) = \beta \\ 0\,, & \text{otherwise} \end{cases} \\
&= \delta(\beta = \gamma + (0,0,1,1)) \\
&= \delta(\beta = \alpha + (-1,-1,1,1)) \\
&= \delta(\alpha = \beta + (1,1,-1,-1))\,.
\end{aligned}$$

Meanwhile $\mathrm{LT}(g_2)|w^\alpha \iff \alpha = \gamma + (0,2,1,1)$ for some γ, so

$$\begin{aligned}
\tau_{\alpha,\beta,2} &= \left(\frac{w^\alpha}{b^2cd}(b^2cd - cd^3)\right)_\beta \\
&= \left(w^\alpha - w^\gamma cd^3\right)_\beta \\
&= -(w^{\gamma+(0,0,1,3)})_\beta \\
&= -\delta(\beta = \gamma + (0,0,1,3)) \\
&= -\delta(\beta = \alpha + (0,-2,0,2)) \\
&= -\delta(\alpha = \beta + (0,2,0,-2))\,.
\end{aligned}$$

Lastly, $\mathrm{LT}(g_3)|w^\alpha \iff \alpha = \gamma + (1,0,1,3)$, so

$$\begin{aligned}
\tau_{\alpha,\beta,3} &= \left(\frac{w^\alpha}{acd^3}(acd^3 + bc^2d^2)\right)_\beta \\
&= \delta(\beta = \gamma + (0,1,2,2)) \\
&= \delta(\beta = \alpha + (-1,1,1,-1)) \\
&= \delta(\alpha = \beta + (1,-1,-1,1))\,.
\end{aligned}$$

Adding these together, we get for $\alpha > \beta$

$$\begin{aligned}
\tau_{\alpha,\beta} &= \sum_{j \in D_\alpha} \tau_{\alpha,\beta,j} \\
&= \delta_{1,\alpha}\delta(\alpha = \beta + (1,1,-1,-1)) \\
&\quad - \delta_{2,\alpha}\delta(\alpha = \beta + (0,2,0,-2)) \\
&\quad + \delta_{3,\alpha}\delta(\alpha = \beta + (1,-1,-1,1))\,,
\end{aligned} \tag{4.1.3}$$

where $\delta_{k,\alpha}$ equals 1 if $\mathrm{LT}(g_k)|w^\alpha$ and 0 otherwise.

We can see that for each $\beta \in \mathbb{Z}^4_{\geq 0}$, there are at most three possible arrows originating from β in the division graph. The only multi-indices α which can have arrows from β are $\beta + v_1$, $\beta + v_2$, and $\beta + v_3$, where $v_1 = (1,1,-1,-1)$, $v_2 = (0,2,0,-2)$ and $v_3 = (1,-1,-1,1)$. Figure 4.1 illustrates these three possible arrows.

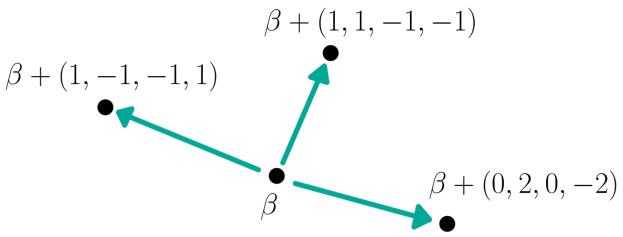


Figure 4.1: In the division graph for the Gröbner basis $G = \{ab + cd, b^2cd - cd^3, acd^3 + bc^2d^2\}$, there are at most three arrows stemming from a multi-index β. These are given by the vectors $v_1 = (1, 1, -1, -1)$, $v_2 = (0, 2, 0, -2)$ and $v_3 = (1, -1, -1, 1)$.

Having found expressions for $\tau_{\alpha,\beta}$, the next step is to compute the Ω^k_α polynomials, where $k = 1,\ 2$ or 3, and indexes over the Gröbner basis $G = \{g_1, g_2, g_3\}$.

Computing Ω^1_α

Because $f_j(w) \propto ab^{2j+1} + cd^{2j+1}$, the set $H_j = \{\alpha \mid (f_j)_\alpha \neq 0\}$ is just $\{(1, 2j+1, 0, 0), (0, 0, 1, 2j+1)\}$, so we only need to compute $\Omega^1_{(1,2j+1,0,0)}$ and $\Omega^1_{(0,0,1,2j+1)}$ for each $j > 0$. These are computed as sums over all paths in the division graph terminating at $(1, 2j+1, 0, 0)$ and $(0, 0, 1, 2j+1)$ respectively. The strategy is to use the expression for $\tau_{\alpha,\beta}$ to find all possible paths leading to these points.

Case 1: $\alpha = (1, 2j+1, 0, 0)$, $w^\alpha = ab^{2j+1}$

$$\begin{aligned} K(\alpha, \alpha) &= \frac{1}{d_\alpha} \\ &= 1 \end{aligned}$$

where $d_\alpha = 1$ since g_1 is the only element of G whose leading term divides ab^{2j+1}. The only indices $\beta < \alpha$ which appear in the expression for Ω^1_α (3.4.1) are those for which $\mathrm{LT}(g_1)|w^\beta$. Any such β will equal $\beta = \gamma + (1, 1, 0, 0)$ for some $\gamma \in \mathbb{Z}^4_{\geq 0}$.

Meanwhile the general expression for $\tau_{\alpha,\beta}$ (4.1.3) tells us that there can only be a path from β to α in the division graph if there exist n_1, n_2, $n_3 \in \mathbb{Z}_{\geq 0}$ such that

$$\alpha = \beta + n_1(1, 1, -1, -1) + n_2(0, 2, 0, -2) + n_3(1, -1, -1, 1)\,.$$

Substituting $\alpha = (1, 2j+1, 0, 0)$ and $\beta = (1, 1, 0, 0) + \gamma$ gives

$$\gamma = (0, 2j, 0, 0) + n_1(-1, -1, 1, 1) + n_2(0, -2, 0, 2) + n_3(-1, 1, 1, -1)\,.$$

As each component of γ must be non-negative, both n_1 and n_3 must be 0, so the possible values of γ are

$$\gamma = (0, 2j - 2n_2, 0, 2n_2)\,,$$

for $n_2 = 0, 1, ..., j$. The indices β which can have paths to $\alpha = (1, 2j+1, 0, 0)$ are then all of the form

$$\begin{aligned} \beta &= (1, 2j + 1 - 2n_2, 0, 2n_2) \\ &= \alpha - n_2(0, 2, 0, -2)\,, \end{aligned}$$

for $n_2 = 1, 2, ..., j$. The $n_2 = 0$ case has been excluded as we are analysing cases where $\beta < \alpha$.

From (4.1.3) we can see that the only possible path from $\beta = (1, 2j+1-2n_2, 0, 2n_2)$ to $\alpha = (1, 2j+1, 0, 0)$ is made up of n_2 individual steps along $(0, 2, 0, -2)$. No steps can be taken along $(1, 1, -1, -1)$ or $(1, -1, -1, 1)$ as that would increase the first component with no way of decreasing it. Meanwhile, the coefficient associated to a step along this possible path is

$$\tau_{(1,2j+1-2l,0,2l),(1,2j+1-2(l+1),0,2(l+1))} = -\delta_{2,(1,2j+1-2l,0,2l)}\,,$$

but this is zero since $\mathrm{LT}(g_2) = b^2cd$ does not divide $w^{(1,2j+1-2l,0,2l)} = ab^{2j+1-2l}d^{2l}$. We conclude

$$K((1, 2j+1, 0, 0), \beta) = 0\,,$$

for all $\beta < \alpha$. Hence only the $\beta = \alpha$ term is non-zero in Ω^1_α, and so

$$\begin{aligned}\Omega^1_\alpha &= K(\alpha,\alpha)\frac{w^\alpha}{\mathrm{LT}(g_1)}\\ &= \frac{ab^{2j+1}}{ab}\\ &= b^{2j}\,.\end{aligned}$$

Case 2: $\alpha = (0, 0, 1, 2j+1)$, $w^\alpha = cd^{2j+1}$

None of the leading terms of the Gröbner basis elements divide cd^{2j+1} so

$$K(\alpha, \alpha) = 0\,.$$

If $\beta < \alpha$, $K(\alpha, \beta)$ only appears in the expression for Ω^1_α if $\mathrm{LT}(g_1) = ab$ divides w^β. In other words, we need to be able to write $\beta = \gamma + (1, 1, 0, 0)$ for some $\gamma \in \mathbb{Z}^4_{>0}$. Additionally, for a path from β to $\alpha = (0, 0, 1, 2j+1)$ to exist in the division graph, there must be $n_1, n_2, n_3 \in \mathbb{Z}_{\geq 0}$ so that

$$\gamma = (-1, -1, 1, 2j+1) + n_1(-1, -1, 1, 1) + n_2(0, -2, 0, 2) + n_3(-1, 1, 1, -1)\,.$$

The right hand side of the above equation always has a negative first component, implying that no such γ exists. Every β which is divisible by $\mathrm{LT}(g_1)$ is not connected by any paths to α. Since $K(\alpha, \beta)$ is a sum over all paths from β to α, we conclude that $K(\alpha, \beta) = 0$ for all $\beta \leq \alpha$ with $\mathrm{LT}(g_1)|w^\beta$, so

$$\Omega^1_{(0,0,1,2j+1)} = 0\,.$$

Convergence: We wish to show convergence of the series $\sum_{j=0}^\infty \left(\sum_{\alpha\in H_j}(f_j)^2_\alpha\right)\left(\sum_{\alpha\in H_j}\left(\Omega^1_\alpha\right)^2\right)$ which in this case is given by

$$\begin{aligned}&\sum_{j=0}^\infty \left(\frac{2}{(2j+1)!}\right)\left((\Omega^1_{(1,2j+1,0,0)})^2 + (\Omega^1_{(0,0,1,2j+1)})^2\right)\\ &\qquad = \sum_{j=0}^\infty \frac{2}{(2j+1)!}b^{4j}\,.\end{aligned}$$

It is straightforward to use the ratio test to check this converges for all values of b. In fact it converges to a continuous function of w. Using the Weierstrass M test, the above series converges uniformly (as a function of w) on any compact subset $U \subset \mathbb{R}^4$. Combining this with the uniform limit theorem tells us that the limit is a continuous function on U, and since U was arbitrary, the limit is continuous on all of $\mathbb{R}^4$.

Computing Ω^2_α

Case 1: $\alpha = (1, 2j+1, 0, 0)$

The relevant indices β for computing Ω^2_α are those which are divisible by $\mathrm{LT}(g_2)$. That is, those of the form $\beta = \gamma + (0, 2, 1, 1)$. For there to be a path from $\gamma + (0, 2, 1, 1)$ to α in the division graph, we require

$$\gamma = (1, 2j-1, -1, -1) + n_1(-1, -1, 1, 1) + n_2(0, -2, 0, 2) + n_3(-1, 1, 1, -1)\,.$$

In order for both the first and third components to be non-negative, exactly one of n_1 or n_3 must equal 1 and the other must equal 0. It appears that there are two types of points β which may have

paths to α in the division graph. The first type of point has $n_1 = 1$, $n_2 \in \{0, 1, ..., j-1\}$, and $n_3 = 0$, and would equal

$$\beta = \alpha - v_1 - n_2 v_2 \,.$$

The other type of point has $n_1 = 0$, $n_2 \in \{1, ..., j\}$, and $n_3 = 1$, equalling

$$\beta = \alpha - n_2 v_2 - v_3 \,.$$

However

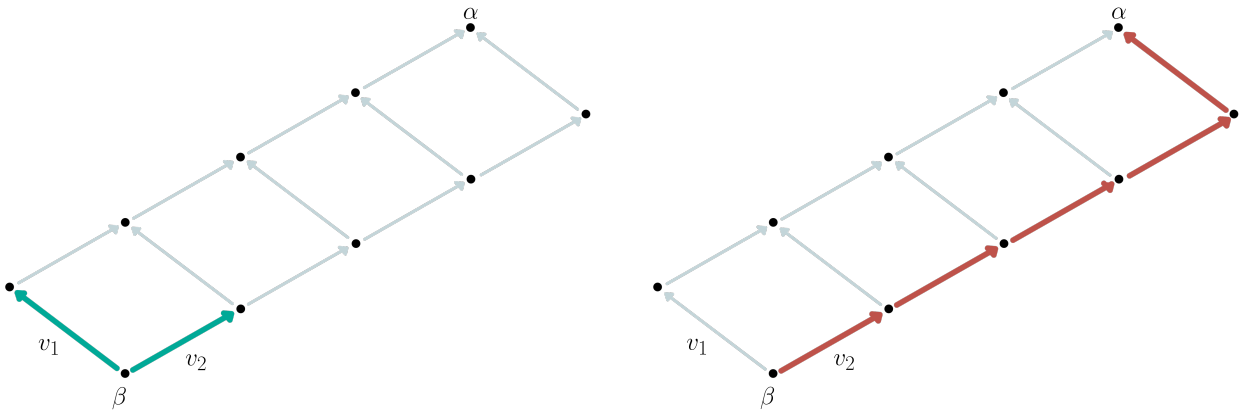
$$v_1 + (n_2 - 1)v_2 = v_3 + n_2 v_2 \,,$$

and so every point from the second case is also covered by the first case. Hence we may assume $n_1 = 1$ and $n_3 = 0$, and the values of β which have paths to $\alpha = (1, 2j+1, 0, 0)$ are

$$\begin{aligned} \beta_{n_2} &= (1, 2j+1, 0, 0) - 1(1, 1, -1, -1) - n_2(0, 2, 0, -2) \\ &= \alpha - v_1 - n_2 v_2 \,, \end{aligned}$$

for $n_2 \in \{0, ..., j-1\}$.

To construct a path from β_{n_2} to α, we need to take one step in the v_1 direction and n_2 steps in the v_2 direction. At first glance, it appears that these steps could be taken in any order, and all the different paths are illustrated in Figure 4.2a. It turns out that only one of these paths is valid.



(a) (b)

Figure 4.2: Figure (a) shows the potential paths from β to α in the division graph, which consist of n_2 steps in the v_2 direction and one step in the v_1 direction. Only one of these paths is valid though. Figure (b) shows this valid path, which is n_2 steps in the v_2 direction followed by a single step in the v_1 direction.

The following condition lets us discard all but one of these potential paths. For a step to be taken from an index η along the v_i direction, $\mathrm{LT}(g_i)$ must divide the end point $\eta + v_i$. We denote a potential path by a sequence of indices $\eta_0 < \eta_1 < ... < \eta_{n_2+1}$, where η_i is the index after i steps have been taken, so $\eta_0 = \beta_{n_2}$ and $\eta_{n_2+1} = \alpha$. Taking a step along the v_1 direction reduces the third component by 1. The index β_{n_2} starts with a third component of 1, and taking steps along the v_2 direction leaves this component unchanged. Hence if $\eta_l \to \eta_{l+1}$ is the step taken along the v_1 direction, then the third component of η_{l+1} will be 0. Taking a step along v_2 would leave this component as zero and $\mathrm{LT}(g_2) = b^2cd$ would not divide the end point of this step. Hence, no steps along v_2 are allowed once the step along v_1 has been taken.

We conclude that the only path from β to α in the division graph consists of taking n_2 steps in the v_2 direction followed by one step in the v_1 direction. This path is illustrated in Figure 4.2b (it is straightforward to check that each step in this path is in fact allowed.) We can write the l^{th} index in

the path as $\eta_l = (0, 2j - 2n_2 + 2l, 1, 1 + 2n_2 - 2l)$ for $l = 0, 1, ..., n_2$, and $\eta_{n_2+1} = \alpha = (1, 2j + 1, 0, 0)$. We should note that there are no paths involving v_3 since taking a step along this direction takes us to an index whose third component is 0, which is not divisible by $\mathrm{LT}(g_3) = acd^3$.

The coefficient $\tau_{\eta_{n_2+2},\eta_{n_2+1}}$ from (4.1.3) associated to the step along v_1 is 1, while the coefficient for a step in the v_2 direction τ_{η_{l+1},η_l} is -1. Of all the leading terms in the Gröbner basis, $\mathrm{LT}(g_1)$, $\mathrm{LT}(g_2)$, and $\mathrm{LT}(g_3)$, only one divides each index in the path η_l, so $d_{\eta_l} = 1$. We compute

$$\begin{aligned} K(\alpha, \beta_{n_2}) &= \frac{(-1)^{n_2+3}}{1} 1 (-1)^{n_2} \\ &= -1 \,. \end{aligned}$$

To summarise, we have found which β are both divisible by $\mathrm{LT}(g_2)$ and have paths in the division graph connecting them to $\alpha = (1, 2j+1, 0, 0)$. We've shown that each of these β's has exactly one path to α and the constant associated to this path is $K(\alpha, \beta) = -1$. This lets us conclude that for $j \geq 1$

$$\begin{aligned} \Omega^2_{(1,2j+1,0,0)} &= \sum_{\substack{\beta \leq \alpha \\ 2 \in D_\beta}} K((1, 2j+1, 0, 0), \beta) \frac{w^\beta}{b^2 cd} \\ &= \sum_{n_2=0}^{j-1} K((1, 2j+1, 0, 0), (0, 2j - 2n_2, 1, 1 + 2n_2)) \frac{b^{2j-2n_2} c d^{1+2n_2}}{b^2 cd} \\ &= \sum_{n_2=0}^{j-1} -b^{2j-2(n_2+1)} d^{2n_2} \,. \end{aligned}$$

Case 2: $\alpha = (0, 0, 1, 2j + 1)$

Any index β which is divisible by $\mathrm{LT}(g_2)$ will equal $\beta = \gamma + (0, 2, 1, 1)$, and for a path to exist from β to α, we must also have

$$\gamma = (0, -2, 0, 2j) - n_1(1, 1, -1, -1) - n_2(0, 2, 0, -2) - n_3(1, -1, -1, 1) \,,$$

but this is impossible. The first or second component will be negative regardless of the values of n_1, n_2, and n_3, so there are no such β with paths to α. $\mathrm{LT}(g_2)$ also doesn't divide α, so $K(\alpha, \alpha)$ doesn't appear in the expression for Ω^2_α. Hence

$$\Omega^2_{(0,0,1,2j+1)} = 0 \,.$$

Convergence: We need to check that the following series converges

$$\begin{aligned} &\sum_{j=0}^{\infty} \frac{2}{(2j+1)!} \left(\left(\Omega^2_{(1,2j+1,0,0)} \right)^2 + \left(\Omega^2_{(0,0,1,2j+1)} \right)^2 \right) \\ &\quad = \sum_{j=0}^{\infty} \frac{2}{(2j+1)!} \left(\sum_{n_2=0}^{j-1} -b^{2j-2(n_2+1)} d^{2n_2} \right)^2 . \end{aligned} \tag{4.1.4}$$

Replacing both b and d with some $s > 0$ gives the series

$$\sum_{j=0}^{\infty} \frac{2}{(2j+1)!} j^2 s^{4j-4} \,.$$

Applying the ratio test, this converges for any value of s. It follows that (4.1.4) converges for all values of b and d in $\mathbb{R}$. The Weierstrass M test and uniform limit theorem again show that the limit is a continuous function on all of $\mathbb{R}^4$.

Computing Ω^3_α

Case 1: $\alpha = (1, 2j+1, 0, 0)$

$\mathrm{LT}(g_3) = acd^3$ does not divide α, so the $K(\alpha, \alpha)$ term does not contribute. For $\mathrm{LT}(g_3)$ to divide an index β, we need that $\beta = \gamma + (1, 0, 1, 3)$. For there to be a path from such β to α we must be able to write

$$\gamma = (0, 2j+1, -1, -3) - n_1(1, 1, -1, -1) - n_2(0, 2, 0, -2) - n_3(1, -1, -1, 1)\,,$$

but this isn't possible. Both of n_1 and n_2 must be zero to prevent the first component from being negative but this will leave the third component negative. We conclude that

$$\Omega^3_{(1,2j+1,0,0)} = 0\,. \tag{4.1.5}$$

Case 2: $\alpha = (0, 0, 1, 2j+1)$

Again the $K(\alpha, \alpha)$ term doesn't contribute. Following the same argument again, we need to be able to write

$$\gamma = (-1, 0, 0, 2j-2) - n_1(1, 1, -1, -1) - n_2(0, 2, 0, -2) - n_3(1, -1, -1, 1)\,, \tag{4.1.6}$$

which again is impossible due to the first component always being negative. We conclude that

$$\Omega^3_{(0,0,1,2j+1)} = 0\,. \tag{4.1.7}$$

Since both $\Omega^3_{(1,2j+1,0,0)}$ and $\Omega^3_{(0,0,1,2j+1)}$ equal 0, there are no series to check convergence for.

4.1.2 Example 1 satisfies (C4)

For each $k = 1, 2, 3$ the series $\sum_{j=0}^{\infty} \left(a_j^k(w)\right)^2$ converges to a continuous function on $\mathbb{R}^4$. This function is bounded on any given compact weight space $W \subset \mathbb{R}^4$, so this model will satisfy conditions (C1) - (C4) on this weight space. Hence there exists $C > 0$ so that

$$K(w) \leq C\left(g_1(w)^2 + g_2(w)^2 + g_3(w)^2\right) \tag{4.1.8}$$

for all $w \in W$.

4.2 Example 2: Two neuron tanh network

The previous example is simpler to study because the activation function $\sin(x)$ has a globally convergent Taylor series. The tanh activation does not have this property, and care must be taken to keep track of its domain of convergence. In this example, we study a two layer tanh network with two neurons and no biases, given by

$$f(x, a, b, c, d) = a\tanh(bx) + c\tanh(dx)\,. \tag{4.2.1}$$

In Section 2.3 we found that when $|bx|, |dx| < \frac{\pi}{2}$, the Taylor series about $x = 0$ for the network f can be written

$$f(x, a, b, c, d) = \sum_{j=0}^{\infty} f_j(a, b, c, d) e_j(x)\,,$$

where

$$f_j(a, b, c, d) = \gamma_j(ab^{2j+1} + cd^{2j+1})\,,$$
$$e_j(x) = \eta_j x^{2j+1}\,,$$

and γ_j and η_j are real numbers satisfying $\gamma_j \eta_j = \frac{2^{2(j+1)}(2^{2(j+1)}-1)B_{2(j+1)}}{(2(j+1))!}$. There is some choice available in defining γ_j and η_j. However, we need to define them in a way which guarantees the convergence of the following series

$$\sum_{j=0}^{\infty} \|e_j(x)\|^2 < \infty \qquad \text{for (C2)}\,, \tag{4.2.2}$$

$$\sum_{j=0}^{\infty} f_j(w)^2 < \infty \qquad \text{for (C3)}\,, \tag{4.2.3}$$

$$\sum_{j=0}^{\infty} \left(\sum_{\alpha \in G_i} (f_j)_\alpha^2 \right) \left(\sum_{\alpha \in G_j} (\Omega_\alpha^k)^2 \right) < \infty \qquad \text{for (C4)}\,. \tag{4.2.4}$$

In the third inequality (4.2.4), k indexes a Gröbner basis $\{g_1, ..., g_s\}$ for the ideal generated by the f_j's.

Condition (C2)

We set the input distribution $q(x)$ to be uniform on $[-t, t]$, for some $t > 0$. We we will investigate what values t, η_j, and γ_j may take while ensuring conditions (C2)-(C4) hold. There is an important conceptial difference between these choices though. The choices of η_j and γ_j don't alter the model under consideration, while adjusting the value of t changes the input space, and hence also changes the model.

To see that some care must be taken in the choice of γ_j and η_j, consider the following definitions which shift "most" of the value in the Taylor series coefficients to the $e_j(x)$'s

$$\gamma_j = \frac{1}{\sqrt{(2(j+1))!}} \qquad \text{and} \qquad \eta_j = \frac{2^{2(j+1)}(2^{2(j+1)}-1)B_{2(j+1)}}{\sqrt{(2(j+1))!}}\,.$$

Factoring the constants this way makes the $f_j(w)$'s almost identical to the ones in the two neuron sine network and condition (C3) automatically holds. Unfortunately this choice makes condition (C2) fail. To see why, we check the convergence of

$$\begin{aligned} \sum_{j=0}^{\infty} \|e_j(x)\|_{L^2(X,q)}^2 &= \sum_{j=0}^{\infty} \int_{[-t,t]} \eta_j^2 x^{4j+2} q(x) dx \\ &= \sum_{j=0}^{\infty} \frac{\eta_j^2}{2t} \frac{2t^{4j+3}}{4j+3}\,. \end{aligned}$$

Setting $q_j = \frac{\eta_j^2}{2t} \frac{2t^{4j+3}}{4j+3}$ and using the ratio test we check

$$\lim_{j \to \infty} \frac{q_{j+1}}{q_j}\,.$$

It's well known that

$$|B_{2n}| \sim 4\sqrt{\pi n} \left(\frac{n}{\pi e} \right)^{2n} \qquad \text{as } n \to \infty\,.$$

So

$$\begin{aligned} \frac{B_{2(j+2)}^2}{B_{2(j+1)}^2} &\sim \frac{(j+2)^{4(j+2)}(\pi e)^{4(j+1)}}{(j+1)^{4(j+1)}(\pi e)^{4(j+2)}} \\ &\sim \frac{1}{(\pi e)^4} \frac{j^{4j+8}\left(1+\frac{2}{j}\right)^{4j+8}}{j^{4j+4}\left(1+\frac{1}{j}\right)^{4j+4}} \end{aligned}$$

$$\sim \frac{j^4}{\pi^4}\,.$$

Meanwhile

$$\frac{2^{4(j+2)}(2^{2(j+2)}-1)^2}{2^{4(j+1)}(2^{2(j+1)}-1)^2} \sim 2^8\,,$$

and

$$\left(\frac{t^{4(j+1)+2}}{t^{4j+2}}\right)\left(\frac{4j+3}{4(j+1)+3}\right)\left(\frac{(2(j+1))!}{(2(j+2))!}\right) \sim \frac{t^4}{(2j+4)(2j+3)}\,.$$

Combining the above results gives

$$\lim_{j\to\infty}\frac{q_{j+1}}{q_j} = \lim_{j\to\infty}\frac{2^8 t^4 j^4}{\pi^4(2j+4)(2j+3)}\,, \tag{4.2.5}$$

which diverges regardless of how small t is, and so for this choice of constants, condition (C2) isn't satisfied.

To have any hope of the $e_j(x)$'s being square summable, we need to remove a factor of j^2 from the numerator of (4.2.5). One way to achieve this is by setting

$$\gamma_j = \frac{\sqrt{|B_{2(j+1)}|}}{\sqrt{(2(j+1))!}} \qquad \text{and} \qquad \eta_j = \frac{2^{2(j+1)}(2^{2(j+1)}-1)\,\mathrm{sgn}(B_{2(j+1)})\sqrt{|B_{2(j+1)}|}}{\sqrt{(2(j+1))!}}\,. \tag{4.2.6}$$

Lemma 12. If the input distribution is the uniform distribution on $[-t,t]$ for any $t < \left(\frac{\pi^2}{2^6}\right)^{\frac{1}{4}}$, then with η_j given in (4.2.6), and $e_j(x) = \eta_j x^{2j+1}$

$$\sum_{j=0}^{\infty} \|e_j(x)\|^2_{L^2(X,q)} \;<\; \infty\,.$$

This is to say, condition (C2) is satisfied.

Proof: Using these definitions we see that

$$\begin{aligned}\frac{q_{j+1}}{q_j} &\sim \frac{2^8 t^4 j^2}{\pi^2(2j+4)(2j+3)}\\ &\sim \frac{2^8 t^4}{4\pi^2}\\ &\sim \frac{2^6 t^4}{\pi^2}\,.\end{aligned} \tag{4.2.7}$$

We can make t necessarily small to ensure convergence of $\sum_{j=0}^{\infty}\|e_j(x)\|^2_{L^2(X,q)}$, in particular if $t < \frac{\sqrt{\pi}}{2\sqrt{2}} \approx 0.62$, the $e_j(x)$'s will be square summable. □

The fact that the input space must be sufficiently small in order for (C2) to hold is unsurprising. Because we are using the Taylor series, it makes sense that its radius of convergence will impose some restrictions on the input space.

Condition (C3)

Next we check if condition (C3) holds with $f_j(w) = \gamma_j(ab^{2j+1} + cd^{2j+1})$, where γ_j is given in (4.2.6).

$$\|f_j(a,b,c,d)\|^2_{\ell^2(\mathbb{R})} = \sum_{j=0}^{\infty}\frac{|B_{2(j+1)}|}{(2(j+1))!}\left(ab^{2j+1}+cd^{2j+1}\right)^2\,.$$

Suppose we restrict the weight (a, b, c, d) so that $|a|,\ |c| < s$ and $|b|,\ |d| < r$ for some $s,\ r > 0$. Then

$$\begin{aligned} |ab^{2j+1} + cd^{2j+1}| &\leq |ab^{2j+1}| + |cd^{2j+1}| \\ &\leq 2sr^{2j+1}\,. \end{aligned}$$

It follows that

$$\|f_j(a, b, c, d)\|_{\ell^2(\mathbb{R})^2} \leq 4s^2 \sum_{j=0}^{\infty} \frac{|B_{2(j+1)}|}{(2(j+1))!} r^{4j+2}\,.$$

Lemma 13. If $|r| < \sqrt{2\pi}$ then

$$\sum_{j=0}^{\infty} \frac{|B_{2(j+1)}|}{(2(j+1))!} r^{4j+2} < \infty\,.$$

Proof: We prove this using the ratio test. Setting $h_j = \frac{|B_{2(j+1)}|}{(2(j+1))!} r^{4j+2}$, we have that

$$\begin{aligned} \lim_{j\to\infty} \frac{h_{j+1}}{h_j} &= \lim_{j\to\infty} \frac{j^2}{\pi^2} \frac{2}{(2j+4)(2j+3)} r^4 \\ &= \lim_{j\to\infty} \frac{r^4}{4\pi^2}\,, \end{aligned}$$

which is less than one precisely when $r < \sqrt{2\pi}$. □

Hence $\|f_j(a, b, c, d)\|_{\ell^2(\mathbb{R})} < \infty$ as long as $|b|, |d| < \sqrt{2\pi}$. The other weights a and c may take any values.

Condition (C4)

Since the f_j's are rescaled versions of the ones in Section 4.1, the set $G = \{g_1, g_2, g_3\} = \{ab + cd, b^2cd - cd^3, acd^3 + bc^2d^2\}$ still works as a Gröbner basis for this example. The Ω_α^k polynomials depend only on the choice of Gröbner basis, and hence are exactly the same as the ones for the two neuron sine network.

Using the values of the Ω_α^k's from Section 4.1, proving condition (C4) reduces to showing the two following series converge and are bounded on the weight space

$$\sum_{j=0}^{\infty} 2\gamma_j^2 b^{4j}\,, \tag{4.2.8}$$

$$\sum_{j=1}^{\infty} 2\gamma_j^2 \left(\sum_{n=0}^{j-1} -b^{2j-2(n+1)} d^{2n} \right)^2\,. \tag{4.2.9}$$

For the first series, (4.2.8), we check the convergence of

$$\sum_{j=0}^{\infty} 2 \frac{|B_{(2(j+1))}|}{(2(j+1))!} b^{4j}\,.$$

By Lemma 13, this converges when $|b| < \sqrt{2\pi}$.

For the second series (4.2.9) we set $b = d = s$ and check the convergence of

$$\sum_{j=1}^{\infty} 2\gamma_j^2 j^2 s^{4j-4}\,.$$

Using the ratio test we let $p_j = 2\gamma_j^2 j^2 s^{4j-4}$, then

$$\begin{aligned}\frac{p_{j+1}}{p_j} &\sim \frac{(j+1)^2 s^{4j}}{4\pi^2 j^2 s^{4j-4}} \\ &\sim \frac{s^4}{4\pi^2}\,,\end{aligned}$$

so we can conclude (4.2.9) converges for all $(b,d) \in (-\sqrt{2\pi}, \sqrt{2\pi}) \times (-\sqrt{2\pi}, \sqrt{2\pi})$. Using the Weierstrass M test, both series converge to continuous functions on $(-\sqrt{2\pi}, \sqrt{2\pi}) \times (-\sqrt{2\pi}, \sqrt{2\pi})$, and so are bounded on any compact subsets.

We lastly must check for which values of x and w the original Taylor of f converges. Inputs and weights must be restricted to ensure $|bx| < \frac{\pi}{2}$ and $|dx| < \frac{\pi}{2}$. This turns out to add no further restrictions. Since we already require $|x| < t < \left(\frac{\pi^2}{2^6}\right)^{(\frac{1}{4})}$ and $|b|$, $|d| < \sqrt{2\pi}$, it follows

$$\begin{aligned}|bx|,\, |dx| &< \sqrt{2\pi}\left(\frac{\pi^2}{2^6}\right)^{(\frac{1}{4})} \\ &= \frac{\pi}{2}\,.\end{aligned}$$

Definition: A *working set* for the neural network f is a subset of the form $U \times (-t,t) \subset W \times \mathbb{R}$ satisfying:

1. $|bx|$, $|dx| < \frac{\pi}{2}$ for all $(a,b,c,d) \in U$ and x in $(-t,t)$.
2. The e_j's are square summable when the input distribution $q(x)$ is defined to be uniform on $(-t,t)$.
3. The two series (4.2.8) and (4.2.9) are bounded as functions on U.

The first condition ensures that the Taylor series of the network converges, while the second and third guarantee conditions (C3) and (C4) respectively.

Roughly speaking a working set is a set on which we can be confident all our methods work and the upper bound obtained is valid. Specifically, if $U \times (-t,t)$ is a working set for the two neuron network f and if the true regression function is $f_T = 0$, then there exists $C > 0$ such that

$$K(a,b,c,d) \leq C\left((ab+cd)^2 + (b^2cd - cd^3)^2\right) \tag{4.2.10}$$

for all $(a,b,c,d) \in U$.

Remark: We have shown that

$$(\mathbb{R} \times [-s,s] \times \mathbb{R} \times [-s,s]) \times [-t,t] \tag{4.2.11}$$

is a working set for f, for any $0 < s < \sqrt{2\pi}$ and $0 < t < \frac{\sqrt{\pi}}{2\sqrt{2}}$.

4.3 Gröbner methods are too difficult for bigger networks

Having used the generic division algorithm to prove the upper bound for a two neuron network, we would like to advance to more complicated examples. However, for this simple example though, computing the quotients $a_j^k(w)$ with the generic division algorithm was not straightforward. Unfortunately this gets much too complicated for larger networks.

Take the two layer tanh network with three neurons

$$g(x,a,,c,d,e,f) = a\tanh(bx) + c\tanh(dx) + e\tanh(fx)$$

which has Taylor series polynomials proportional to

$$g_j(w) = ab^{2j+1} + cd^{2j+1} + ef^{2j+1}\,.$$

Using the computer algebra system Singular [DGPS21] with graded lexicographic monomial order we find the following Gröbner basis for the ideal $I = \langle \{g_j\}_{j=0}^{\infty} \rangle$

$$\begin{aligned} G = \Big\{ & ab + cd + ef, b^2cd + b^2ef - cd^3 - ef^3, acd^3 + aef^3 + bc^2d^2 + 2bcdef + be^2f^2, \\ & b^2d^2ef - b^2ef^3 - d^2ef^3 + ef^5, b^2cef^3 + b^2de^2f^2 - cd^4ef + cd^2ef^3 - cef^5 - de^2f^4, \\ & ad^2ef^3 - aef^5 + bcd^3ef - bcdef^3 + bd^2e^2f^2 - be^2f^4, cd^5ef - 2cd^3ef^3 + cdef^5, \\ & acdef^5 + ae^2f^6 - bc^2d^4ef + 2bc^2d^2ef^3 - bcd^3e^2f^2 + 3bcde^2f^4 + be^3f^5, \\ & acef^7 + ade^2f^6 - bc^2d^3ef^3 + 2bc^2def^5 - bcd^4e^2f^2 + 2bcd^2e^2f^4 + bce^2f^6 + bde^3f^5 \Big\}\,. \end{aligned}$$

Not only does this basis have nine elements, but several of these elements contain many terms. For each basis element g_k, we have to check the convergence of the associated series of quotients $\sum_{j=0}^{\infty} a_j^k(w)^2$, so the larger the basis, the longer the computation. Additionally, if a basis polynomial g_k has many terms then so will

$$\frac{w^\alpha}{\mathrm{LT}(g_k)} g_k\,,$$

for each w^α divisible by $\mathrm{LT}(g_k)$. This makes more of the $\tau_{\alpha,\beta}$'s in the division graph non-zero, resulting in a division graph containing many more arrows. As a result, there are many more paths between each pair of indices α and β, making the calculation much more difficult.

For us, the division graph method was too difficult for any network with three neurons or more. This leaves few options for finding quotients for larger networks. One possibility is to take each Taylor series polynomial f_j and divide it by the basis G using the standard division algorithm. This can be done using a computational algebra package, albeit with one large obstacle: there are infinitely many polynomials f_j to divide. Unless there is some pattern in the definition of the f_j's allowing you to perform a finite number of divisions and extrapolate their results to the rest of the sequence of polynomials, this approach will not work.

We should note that generic Gröbner basis methods of the previous examples cannot avoid this problem either. The calculation was only possible because the f_j's followed a simple pattern, which allowed us to do a single calculation for an arbitrary f_j.

At present then, our best option for studying other examples is to hope we are lucky enough to find a pattern in the quotients. Admittedly this approach is somewhat inelegant and not very generalisable.

4.4 Example 3: n neuron tanh networks

In this example we find polynomial upper bounds for the KL divergence for two layer networks with arbitrarily many neurons. The network to be analysed is given by the function

$$f(x, w) = \sum_{i=1}^{n} a_i \tanh(b_i x)\,. \tag{4.4.1}$$

In Section 2.3 we saw that when $|b_i x| < \frac{\pi}{2}$ for all i, the Taylor series of f gives

$$f(x, w) = \sum_{j=0}^{\infty} f_j(w) e_j(x)\,,$$

where $f_j(w) = \gamma_j \left(\sum_{i=1}^{n} a_i b_i^{2j+1} \right)$, $e_j(x) = \eta_j x^{2j+1}$, and γ_j, η_j are any real numbers satisfying $\gamma_j \eta_j = \frac{2^{2(j+1)}(2^{2(j+1)}-1)B_{2(j+1)}}{(2(j+1))!}$.

We use the same choice of γ_j and η_j from the two neuron example, setting

$$\gamma_j = \frac{\sqrt{|B_{2(j+1)}|}}{\sqrt{(2(j+1))!}} \qquad \text{and} \qquad \eta_j = \frac{2^{2(j+1)}(2^{2(j+1)}-1)\operatorname{sgn}(B_{2(j+1)})\sqrt{|B_{2(j+1)}|}}{\sqrt{(2(j+1))!}}\,.$$

To prove a polynomial upper bound for the KL divergence of this model, we first check conditions (C2) and (C3).

Condition (C2)

The $e_j(x)$'s for the n neuron network are exactly the same as in the two neuron network. Hence the proof of condition (C2) for the two neuron network carries over to this example as long as the input distribution $q(x)$ is the uniform distribution on $[-t,t]$ where $t < \left(\frac{\pi^2}{2^6}\right)^{\frac{1}{4}}$.

Condition (C3)

To check condition (C3), we need to show

$$\sum_{j=0}^{\infty} \frac{|B_{2(j+1)}|}{(2(j+1))!}\left(\sum_{i=1}^{n} a_i b_i^{2j+1}\right)^2 < \infty.$$

We will use a similar technique as from the two neuron example. Suppose $|a_1|, |a_2|, ..., |a_n| \leq s$ and $|b_1|, ..., |b_n| \leq r$ for some $s, r > 0$. Then

$$\begin{aligned}\left(\sum_{i=1}^{n} a_i b_i^{2j+1}\right)^2 &\leq \left(\sum_{i=1}^{n} |a_i||b_i|^{2j+1}\right)^2 \\ &\leq \left(nsr^{2j+1}\right)^2.\end{aligned}$$

Hence

$$\sum_{j=0}^{\infty} f_j(w)^2 \leq n^2 s^2 \sum_{j=0}^{\infty} \frac{|B_{2(j+1)}|}{(2(j+1))!} r^{4j+2}\,.$$

By Lemma 13, the series on the right converges as long as $r < \sqrt{2\pi}$. Hence condition (C3) holds provided $|b_1|, ..., |b_n| < \sqrt{2\pi}$.

The final part of the calculation is condition (C4), which we will split into two sections. The first step is to find a basis for the ideal $I = \langle \{f_j(w)\}_{j=0}^{\infty} \rangle$ and compute the quotients $a_j^k(w)$ for this basis. The second step is to show that the sum of squares of these quotients form a bounded series.

4.4.1 Condition (C4): Finding a basis and quotients

For the purpose of finding a basis of the ideal $I = \langle \{f_j(w)^2\}_{j=0}^{\infty} \rangle$, we can rescale the f_j's so that each term has coefficent 1. For this section, we drop γ_j and set

$$f_j(w) = \sum_{i=1}^{n} a_i b_i^{2j+1}\,.$$

When we reach the final step of checking convergence, we will reincorporate the γ_j's.

The problem of finding a basis for the ideal $I = \langle \{f_j(w)\}_{j=0}^{\infty} \rangle \subset \mathbb{R}[a_1, ..., a_n, b_1, ..., b_n]$ has been solved by Takahashi and Washino. They found a recursive expression for the f_j's, to show that

$$B = \{f_0, f_1, ..., f_{n-1}\}$$

is a basis for I [TW20]. In particular, they derived the expression

$$\begin{aligned} f_{j+n-1} =& f_{n-1}\left(\sum_{i=1}^{n} b_i^{2j}\right) + f_{n-2}\left(\sum_{i=1}^{n}\sum_{k\neq i} b_i^{2j}b_k^2\right) + f_{n-3}\left(\sum_{i=1}^{n}\sum_{k\neq i}\sum_{l>k,l\neq i} b_i^{2j}b_k^2b_l^2\right) + \\ & \ldots + f_0\left(\sum_{i=1^n} b_i^{2j}\prod_{k\neq i} b_k^2\right) \\ & + f_{j+n-3}\left(\sum_{\substack{i_1,i_2 \\ i_t\neq i_q \text{ for } t\neq q}} b_{i_1}^2 b_{i_2}^2\right) - 2f_{j+n-4}\left(\sum_{\substack{i_1,i_2,i_3 \\ i_t\neq i_q \text{ for } t\neq q}} b_{i_1}^2 b_{i_2}^2 b_{i_3}^2\right) + \\ & \ldots + (-1)^n(n-1)f_{j-1}\left(b_1^2b_2^2\ldots b_n^2\right) . \end{aligned}$$

We first tried to "unravel" this recursive statement to find expressions for each f_j in terms of only $f_0, \ldots, f_{n-1}$ but this was too difficult even for small three neuron networks.

Instead, we used a computer algebra package to compute quotients for three and four neurons, before generalising the results to networks with arbitrarily many neurons.

Three Neurons

The Taylor series polynomials for a two layer three neuron network are

$$p_j(w) = ab^{2j+1} + cd^{2j+1} + ef^{2j+1} .$$

By Takahashi and Washino's result, $\langle p_0, p_1, p_2\rangle = \langle\{p_j\}_{j=0}^{\infty}\rangle$. We used Singular to divide the first few p_j polynomials by the basis (p_0, p_1, p_2) with graded lexicographic order. This output the following expressions

$$p_3 = \left(b^2d^2f^2\right)p_0 - \left(b^2d^2 + b^2f^2 + d^2f^2\right)p_1 + \left(b^2 + d^2 + f^2\right)p_2$$

$$\begin{aligned} p_4 =& \left(b^4d^2f^2 + b^2d^4f^2 + b^2d^2f^4\right)p_0 - \left(b^4d^2 + b^4f^2 + b^2d^4 + 2b^2d^2f^2 + b^2f^4 + d^4f^2 + d^2f^4\right)p_1 \\ & + \left(b^4 + b^2d^2 + b^2f^2 + d^4 + d^2f^2 + f^4\right)p_2 \end{aligned}$$

$$\begin{aligned} p_5 =& \left(b^6d^2f^2 + b^4d^4f^2 + b^4d^2f^4 + b^2d^6f^2 + b^2d^4f^4 + b^2d^2f^6\right)p_0 \\ & - \left(b^6d^2 + b^6f^2 + b^4d^4 + 2b^4d^2f^2 + b^4f^4 + b^2d^6 + 2b^2d^4f^2 + 2b^2d^2f^4 + b^2f^6 + d^6f^2 + \right. \\ & \left. d^4f^4 + d^2f^6\right)p_1 \\ & + \left(b^6 + b^4d^2 + b^4f^2 + b^2d^4 + b^2d^2f^2 + b^2f^4 + d^6 + d^4f^2 + d^2f^4 + f^6\right)p_2 . \end{aligned}$$

A pattern begins to emerge. The quotient of p_0 in the expression for p_j is made up of monomials containing all three of the variables b, d, and f. Each of these variables is put to an even power and the total degree of each of these monomials equals $2j$. The quotient of p_1 consists of monomials containing either 2 or 3 of the variables b, d, and e, again each with even powers, and the total degree of each monomial is $2j-2$. When one of these monomials contains all three of the variables b, d, and f, it is paired with a coefficient of 2, while the monomials in only two variables have coefficient 1. Lastly, the quotient of p_2 contains all monomials in 1, 2, or 3 of the variables, again made up of even powers, with total degree $2j-4$. The coefficients of each monomial in this expression is just 1. This suggests that an arbitrary p_j can be written in the following form.

Proposition 4. For $j > 2$, p_j can be written as

$$p_j = \left(\sum_{\substack{|i|=j-2 \\ n_0(i)=2}} \left(b^{i_1}d^{i_2}f^{i_3}\right)^2 + \sum_{\substack{|i|=j-2 \\ n_0(i)=1}} \left(b^{i_1}d^{i_2}f^{i_3}\right)^2 + \sum_{\substack{|i|=j-2 \\ n_0(i)=0}} \left(b^{i_1}d^{i_2}f^{i_3}\right)^2\right)\left(ab^5 + cd^5 + ef^5\right)$$

$$
- \left(\sum_{\substack{|i|=j-1 \\ n_0(i)=1}} \left(b^{i_1} d^{i_2} f^{i_3}\right)^2 + 2 \sum_{\substack{|i|=j-1 \\ n_0(i)=0}} \left(b^{i_1} d^{i_2} f^{i_3}\right)^2 \right) \left(ab^3 + cd^3 + ef^3\right)
$$

$$
\sum_{\substack{i_1+i_2+i_3=j \\ n_0(i_1,i_2,i_3)=0}} \left(b^{i_1} d^{i_2} f^{i_3}\right)^2 \left(ab + cd + ef\right) \,, \tag{4.4.2}
$$

where $i = (i_1, i_2, i_3)$ is a multi-index and $n_0(i)$ counts the number of zero components in i.

This proposition can be proven directly, however we omit this to focus on the equivalent result for networks with more neurons.

Four Neurons

The Taylor series polynomials for a two layer four neuron network are

$$
p_j = ab^{2j+1} + cd^{2j+1} + ef^{2j+1} + gh^{2j+1}
$$

and (p_0, p_1, p_2, p_3) is a basis for the ideal $I = \langle p_0, p_1, p_2, p_3 \rangle$. Using Singular to divide a few of the p_j's by this basis, the following pattern emerges.

Proposition 5. Each $p_j = ab^{2j+1} + cd^{2j+1} + ef^{2j+1} + gh^{2j+1}$ can be written as

$$
\begin{aligned}
p_j = & \left(\sum_{\substack{|i|=j-3 \\ n_0(i)=3}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 + \sum_{\substack{|i|=j-3 \\ n_0(i)=2}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 + \sum_{\substack{|i|=j-3 \\ n_0(i)=1}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 + \sum_{\substack{|i|=j-3 \\ n_0(i)=0}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 \right) p_3 \\
& - \left(\sum_{\substack{|i|=j-2 \\ n_0(i)=2}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 + 2 \sum_{\substack{|i|=j-2 \\ n_0(i)=1}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 + 3 \sum_{\substack{|i|=j-2 \\ n_0(i)=0}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 \right) p_2 \\
& + \left(\sum_{\substack{|i|=j-1 \\ n_0(i)=1}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 + 3 \sum_{\substack{|i|=j-1 \\ n_0(i)=0}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 \right) p_1 \\
& - \left(\sum_{\substack{|i|=j \\ n_0(i)=0}} \left(b^{i_1} d^{i_2} f^{i_3} h^{i_4}\right)^2 \right) p_0 \,,
\end{aligned} \tag{4.4.3}
$$

where $i = (i_1, i_2, i_3, i_4)$ and $|i| = i_1 + i_2 + i_3 + i_4$.

In the above expression for p_j, the basis element p_k is multiplied by the sum of the squares of all monomials of degree $j - k$ in the variables b, d, f, and h, where each monomial contains $4, 3, ..., 4 - k$ of the variables. Each monomial is paired with an integer coefficient, dependent on how how many of the variables it contains. When these monomials are arranged into a grid as in (4.4.3), the pattern for these coefficients becomes clear. The coefficients in the first row are all equal to 1, while the coefficient in the l^{th} row and r^{th} column is given by the sum of the coefficients in previous row $(l-1)$, from the first to the r^{th} column. An alternating sign is then applied to each row.

The expression for p_j in the 3 neuron case (4.4.2) follows the same pattern, suggesting that this construction may work for a general n neuron network.

n Neurons

The Taylor series polynomials for a two layer n neuron network up to some constants are

$$f_j(w) = a_1 b_1^{2j+1} + a_2 b_2^{2j+1} + \cdots + a_n b_n^{2j+1}\,,$$

and Takahashi and Washino's result tells us that $\{f_0, ..., f_{n-1}\}$ is a basis for the ideal generated by all these polynomials.

Definition 25. Define $\mathcal{B}_{1,t} = 1$ for $t = 0, 1, ..., n-1$, and then define recursively $\mathcal{B}_{r,t} = \sum_{k=0}^{t} \mathcal{B}_{r-1,k}$ for $t = 0, ..., n-r$. These integers can be arranged in a grid as shown below in Figure 4.3a, so that the first row consists of 1's, the second row is constructed by summing up elements of the first row, the third row from sums of elements in the second row, and so on. Notably the diagonals of the figure form the rows of Pascal's triangle.

We then define $\mathcal{A}_{r,t} = (-1)^r \mathcal{B}_{r,t}$. So the $\mathcal{A}_{r,t}$'s can be represented by alternating the sign on each row of diagram for the $\mathcal{B}_{r,t}$ values, as shown in Figure 4.3b.

r=1	t=0	1	2	3	4	5
	1	1	1	1	1	1
2	1	2	3	4	5	
3	1	3	6	10		
4	1	4	10			
5	1	5				
6	1					

(a) Grid of $\mathcal{B}_{r,t}$ values for 6 neurons

r=1	t=0	1	2	3	4	5
	1	1	1	1	1	1
2	-1	-2	-3	-4	-5	
3	1	3	6	10		
4	-1	-4	-10			
5	1	5				
6	-1					

(b) Grid of $\mathcal{A}_{r,t}$ values for 6 neurons

Figure 4.3

The following lemma generalises the pattern from the three and four neuron networks to show how any f_j can be written in terms of the basis $\{f_0, ..., f_{n-1}\}$.

Lemma 14. When $j \geq n$, f_j can be written in terms of f_0,..., f_{n-1} as follows

$$\begin{aligned} f_j(w) = &\left(\sum_{t=0}^{n-1} \mathcal{A}_{1,t} \sum_{\substack{|\alpha|=j-(n-1) \\ n_0(\alpha)=n-1-t}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 \right) f_{n-1}(w) \\ &+ \left(\sum_{t=0}^{n-2} \mathcal{A}_{2,t} \sum_{\substack{|\alpha|=j-(n-2) \\ n_0(\alpha)=n-2-t}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 \right) f_{n-2}(w) \\ &\vdots \\ &+ \left(\sum_{t=0}^{0} \mathcal{A}_{n,t} \sum_{\substack{|\alpha|=j \\ n_0(\alpha)=0}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 \right) f_0(w)\,, \end{aligned} \tag{4.4.4}$$

or, more compactly

$$f_j(w) = \sum_{r=1}^{n} \left(\left(\sum_{t=0}^{n-r} \mathcal{A}_{r,t} \sum_{\substack{|\alpha|=j-(n-r) \\ n_0(\alpha)=n-r-t}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 \right) f_{n-r}(w) \right) , \tag{4.4.5}$$

where the sum $\sum_{\substack{|\alpha|=j-(n-r) \\ n_0(\alpha)=n-r-t}}$ is taken over all multi-indices α which satisfy the listed conditions.

In other words, the quotient of the basis element $f_k(w)$ $(0 \leq k \leq n-1)$ in the expression for the Taylor series polynomial f_j $(j \geq 0)$ consists of a sum of squares of several monomials. Specifically, these are the squares of every monomial in the variables $b_1, ..., b_n$ of total degree $j-k$, which contain between $n-k$ and n of the variables. Each monomial is paired with an integer coefficient, which depends on how many of the variables it contains.

Proof: We will use the following notation frequently in this proof. Given a multi-index $\alpha = (\alpha_1, ..., \alpha_n)$, write $\hat{\alpha} := (\alpha_2, ..., \alpha_n)$ for the projection onto all but the first component.

Each of the quotients in equation (4.4.5) are invariant under any transposition of the variables of the form $\sigma_{kl} : \mathbb{R}[a_1, ..., a_n, b_1, ..., b_n]$ which sends $a_k \mapsto a_l$, $a_l \mapsto a_k$, $b_k \mapsto b_l$ and $b_l \mapsto b_k$, while leaving all the other variables unchanged. It follows that if

$$a_1 b_1^{2j+1} = \sum_{r=1}^{n} \left(\left(\sum_{t=0}^{n-r} \mathcal{A}_{r,t} \sum_{\substack{|\alpha|=j-(n-r) \\ n_0(\alpha)=n-r-t}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 \right) a_1 b_1^{2(n-r)+1} \right) \tag{4.4.6}$$

is true, then applying σ_{1k} for each $k = 2, ..., n$ to both sides is sufficient to prove equation (4.4.5).

The overall strategy of the proof is to bring the factors $b_1^{2(n-r)+1}$ in the expressions

$$\left(\left(\sum_{t=0}^{n-r} \mathcal{A}_{r,t} \sum_{\substack{|\alpha|=j-(n-r) \\ n_0(\alpha)=n-r-t}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 \right) a_1 b_1^{2(n-r)+1} \right)$$

into the sums. This lets us change the summation variables, and after doing so, we will see that nearly every term on the right of (4.4.6) cancels out, leaving only $a_1 b_1^{2j+1}$.

Let's consider the expression

$$\sum_{\substack{|\alpha|=j-(n-r) \\ n_0(\alpha)=n-r-t}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 a_1 b_1^{2(n-r)+1} = \sum_{\substack{|\alpha|=j-(n-r) \\ n_0(\alpha)=n-r-t}} (b_1^{\alpha_1+(n-r)} b_2^{\alpha_2} ... b_n^{\alpha_n})^2 a_1 b_1 \tag{4.4.7}$$

for some choice of r and t, which appears in the right hand side of (4.4.6). To change the summation variable, there are two cases to consider based on the value of $n-r-t$.

The first case is when $n-r-t=0$, so that $n_0(\alpha) = 0$, or in other words, each multi-index α in the sum has no components equal to zero. Since the following sets of monomials are equal

$$\left\{ b_1^{\alpha_1+(n-r)} b_2^{\alpha_2} ... b_n^{\alpha_n} \;\middle|\; |\alpha| = j-(n-r), n_0(\alpha) = 0 \right\} = \left\{ b_1^{\alpha_1} ... b_n^{\alpha_n} \;\middle|\; |\alpha| = j,\ \alpha_1 \geq n-r+1,\ n_0(\hat{\alpha}) = 0 \right\},$$

we can rewrite the polynomial in (4.4.7) as

$$\sum_{\substack{|\alpha|=j-(n-r) \\ n_0(\alpha)=n-r-t}} (b_1^{\alpha_1+(n-r)} b_2^{\alpha_2} ... b_n^{\alpha_n})^2 a_1 b_1 = \sum_{\substack{|\alpha|=j \\ \alpha_1 \geq n-r+1 \\ n_0(\hat{\alpha})=0}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 a_1 b_1 \,. \tag{4.4.8}$$

The second case is when $n_0(\alpha) = n - r - t > 0$, meaning that the first component of each multi-index being summed over, α_1, is allowed to be zero or non-zero. In this case,

$$\sum_{\substack{|\alpha|=j-(n-r)\\ n_0(\alpha)=n-r-t}} (b_1^{\alpha_1+(n-r)} b_2^{\alpha_2} \ldots b_n^{\alpha_n})^2 a_1 b_1 = \sum_{\substack{|\alpha|=j-(n-r)\\ \alpha_1=0\\ n_0(\hat{\alpha})=n-r-t-1}} (b_1^{\alpha_1+(n-r)} b_2^{\alpha_2} \ldots b_n^{\alpha_n})^2 a_1 b_1 + \sum_{\substack{|\alpha|=j-(n-r)\\ \alpha_1\geq 1\\ n_0(\hat{\alpha})=n-r-t}} (b_1^{\alpha_1+(n-r)} b_2^{\alpha_2} \ldots b_n^{\alpha_n})^2 a_1 b_1 \,.$$

The first term on the right adds up over all the multi-indices α whose first component α_1 is zero, while the second sum contains all the multi-indices whose first component is non-zero. Applying a change of variables to the summation indices shows that this equals

$$\sum_{\substack{|\alpha|=j\\ \alpha_1=n-r\\ n_0(\hat{\alpha})=n-r-t-1}} (b_1^{\alpha_1} \ldots b_n^{\alpha_n})^2 a_1 b_1 + \sum_{\substack{|\alpha|=j\\ \alpha_1\geq n-r+1\\ n_0(\hat{\alpha})=n-r-t}} (b_1^{\alpha_1} \ldots b_n^{\alpha_n})^2 a_1 b_1 \,. \tag{4.4.9}$$

We can rewrite the right hand side of the proposed equation (4.4.6) by substituting in (4.4.8) and (4.4.9). Doing so gives

$$\begin{aligned} &\sum_{r=1}^{n} \mathcal{A}_{r,n-r} \sum_{\substack{|\alpha|=j\\ \alpha_1\geq n-r+1\\ n_0(\hat{\alpha})=0}} (b_1^{\alpha_1} \ldots b_n^{\alpha_n})^2 a_1 b_1 \\ &+ \sum_{n=1}^{r} \sum_{t=0}^{n-r-1} \mathcal{A}_{r,t} \left(\sum_{\substack{|\alpha|=j\\ \alpha_1=n-r\\ n_0(\hat{\alpha})=n-r-t-1}} (b_1^{\alpha_1} \ldots b_n^{\alpha_n})^2 + \sum_{\substack{|\alpha|=j\\ \alpha_1\geq n-r+1\\ n_0(\hat{\alpha})=n-r-t}} (b_1^{\alpha_1} \ldots b_n^{\alpha_n})^2 \right) a_1 b_1 \,. \end{aligned} \tag{4.4.10}$$

We define the polynomials

$$p_{l,k} = \sum_{\substack{|\alpha|=j\\ \alpha_1=l\\ n_0(\hat{\alpha})=k}} (b_1^{\alpha_1} \ldots b_n^{\alpha_n})^2 a_1 b_1 \tag{4.4.11}$$

for $0 \leq l \leq j$ and $0 \leq k \leq n-1$, and note that $\{p_{l,k} \mid 0 \leq l \leq j, 0 \leq k \leq n-1\}$ is a linearly independent set in $\mathbb{R}[a_1, b_1, ..., a_n, b_n]$. The expression (4.4.10) can then be written in terms of these polynomials as

$$\sum_{r=1}^{n} \mathcal{A}_{r,n-r} \sum_{n-r+1\leq l\leq j} p_{l,0} + \sum_{r=1}^{n} \sum_{t=0}^{n-r-1} \mathcal{A}_{r,t} \left(p_{n-r,n-r-t-1} + \sum_{n-r+1\leq l\leq j} p_{l,n-r-t} \right) . \tag{4.4.12}$$

We begin by finding the coefficients of $p_{s,0}$ in the above expression for a given value of s. The polynomial $p_{s,0}$ appears in $\sum_{n-r+1\leq l\leq j} p_{l,0}$ whenever $s \geq n-r+1$, while it never appears in $\sum_{n-r+1\leq l\leq j} p_{l,n-r-t}$, as $n-r-t$ is always greater than or equal to 1. Hence the coefficient is

$$\left(\sum_{r=n-s+1}^{n} \mathcal{A}_{r,n-r} \right) + A_{n-s,s-1} \,. \tag{4.4.13}$$

Meanwhile, for $0 \leq s < j$ and $0 < h < n-1$, the coefficient of $p_{s,h}$ takes the same form

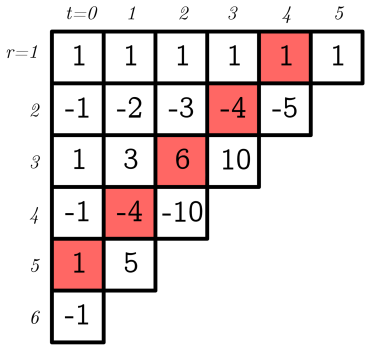
$$\left(\sum_{r=n-s+1}^{n} \mathcal{A}_{r,n-r-h} \right) + \mathcal{A}_{n-s,s-h-1} \,, \tag{4.4.14}$$

where $\mathcal{A}_{l,q}$ is taken to be zero if $l < 1$ or $q < 0$.

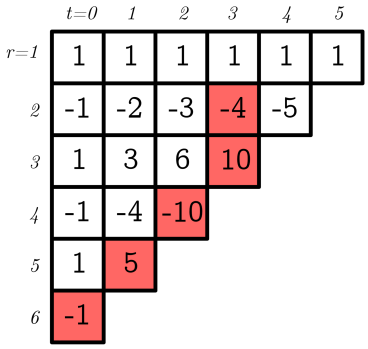
In either case, the sums (4.4.13) and (4.4.14) equal zero. To see why, consider the diagram of Pascal's triangle used to construct the $\mathcal{A}_{r,t}$'s, Figure 4.3b. The sum $\sum_{r=n-s+1}^{n} \mathcal{A}_{r,n-r-h}$ adds up the entries of several boxes in the diagram, starting at the box in row $n-s+1$, column $s-h-1$ and moving diagonally downwards to the left, until column 0 is reached. The box associated to the extra term $\mathcal{A}_{n-s,s-h-1}$ lies directly above the starting point of this diagonal line. There are two possible cases of the shape drawn out in the diagram by these sums.

In the first case, $n-s < 1$ so that the extra $\mathcal{A}_{n-s,s-h-1}$ term lies outside the diagram and is zero. Then the expression (4.4.14) is a sum along a whole diagonal of the associated diagram. This corresponds to adding up a row of Pascals triangle but with alternating signs on each term, which just equals $(1-1)^i$ for some $i > 0$, and hence is 0.

In the second case, $n - s \geq 1$ and the extra term $\mathcal{A}_{n-s,s-h-1}$ is non-zero. The type of path summed over in this case is illustrated in Figure 4.4b.



(a) The first type of summation path is a whole diagonal in the diagram



(b) An example of the second type of summation path

Figure 4.4

To show that the expression in (4.4.14) is zero in this case, we prove the following claim.

Claim: For any l, q

$$\mathcal{A}_{l,q} = \sum_{r=1}^{l} \mathcal{A}_{r,l-r+q+1} \,,$$

or in other words, a single box can be replaced by the box directly to it's right and the diagonal stemming from it going upwards to the right.

Figure 4.5: The claim says that the two shaded regions are equal. This is clearly true for this example.

Proof of claim: Via induction on l. When $l = 1$, $\mathcal{A}_{1,q} = \mathcal{A}_{1,q+1} = 1$, proving the base case.

Suppose the claim holds for some $l > 1$ and all values of q. Then since the $|\mathcal{A}_{i,j}|$'s form pascals triangle, $|\mathcal{A}_{l+1,q+1}| = |\mathcal{A}_{l+1,q}| + |\mathcal{A}_{l,q+1}|$. Meanwhile $\operatorname{sgn}(\mathcal{A}_{l+1,q}) = \operatorname{sgn}(\mathcal{A}_{l+1,q+1}) = -\operatorname{sgn}(\mathcal{A}_{l,q+1})$ and it follows that either $\mathcal{A}_{l+1,q+1} = \mathcal{A}_{l+1,q} - \mathcal{A}_{l,q+1}$ or $-\mathcal{A}_{l+1,q+1} = -\mathcal{A}_{l+1,q} + \mathcal{A}_{l,q+1}$. In either case

$$\mathcal{A}_{l+1,q} = \mathcal{A}_{l+1,q+1} + A_{l,q+1}\,.$$

Applying the induction case to $\mathcal{A}_{l,q+1}$ results in

$$\mathcal{A}_{l+1,q} = \sum_{r=1}^{l+1} \mathcal{A}_{r,l+2-r+q}\,.$$

□

This implies the type of path summed over in the second case is equivalent to summing over a whole diagonal of the diagram, again resulting in 0.

We've found that the coefficient of each of the polynomials $p_{s,h}$ in (4.4.12) is 0 when $0 \leq s < j$ and $0 < h < n-1$. The only remaining polynomial is $p_{j,n-1}$, which has coefficient $\mathcal{A}_{1,0}$ and this equals 1.

Hence, the right hand side of (4.4.6) equals $p_{j_{n-1}}$, but

$$p_{j,n-1} = \sum_{\substack{|\alpha|=j \\ \alpha_1=j \\ n_0(\hat{\alpha})=n-1}} (b_1^{\alpha_1}...b_n^{\alpha_n})^2 a_1 b_1$$

$$= a_1 b_1^{2j+1}\,,$$

which proves the lemma. □

4.4.2 Condition (C4): Convergence

For an n neuron network, we've found a way of writing each Taylor series polynomial f_j in terms of the basis $\{f_0, ..., f_{n-1}\}$. Specifically we've found quotients $a_j^k(w) \in \mathbb{R}[w]$ so that

$$f_j(w) = \sum_{k=0}^{n-1} a_j^k(w) f_k(w)\,.$$

To show the polynomial upper bound of the KL divergence, we now need to show these quotients satisfy condition (C4). We first need to reintroduce the constants from the Taylor series to the $f_j(w)$'s

and $e_j(x)$'s. We set

$$f_j(w) = \sum_{i=1}^{n} a_i b_i^{2j+1} \qquad \text{and} \qquad e_j(x) = \gamma_j \eta_j x^{2j+1}$$

for $0 \leq j \leq n-1$. For values of j greater than $n-1$ we define

$$f_j(w) = \gamma_j \sum_{i=1}^{n} a_i b_i^{2j+1} \qquad \text{and} \qquad e_j(x) = \eta_j x^{2j+1}\,,$$

where

$$\gamma_j = \frac{\sqrt{|B_{2(j+1)}|}}{\sqrt{(2(j+1))!}} \qquad \text{and} \qquad \eta_j = \frac{2^{2(j+1)}(2^{2(j+1)}-1)\,\mathrm{sgn}(B_{2(j+1)})\sqrt{|B_{2(j+1)}|}}{\sqrt{(2(j+1))!}}\,.$$

We've allocated the constants differently between the two cases $0 \leq j \leq n-1$ and $j \geq n$ so that the only adjustment we need to make to the a_j^k's of (4.4.5) is to multiply them by γ_j. This choice ensures that

$$f(x,w) = \sum_{j=0}^{\infty} f_j(w) e_j(x)\,,$$

whenever $|b_1 x|, |b_2 x|, ..., |b_n x| < \frac{\pi}{2}$.

With this choice of constants, we can show condition (C4) holds to prove the following upper bound.

Lemma 15. (Bound for n neuron networks) Let f be a two layer tanh network with n neurons and no biases. Suppose the weight space is restricted to

$$W = (\mathbb{R} \times [-s, s])^n$$

and the input distribution $q(x)$ is the uniform distribution on $[-t, t]$, for some fixed $0 < s < \sqrt{2\pi}$ and $0 < t < \frac{\sqrt{\pi}}{2\sqrt{2}}$. If the true regression function is $f_T(x) = 0$, then there exists $c > 0$ such that

$$K(w) \leq c\left(f_0(w)^2 + ... + f_{n-1}(w)^2\right)$$

for all $w \in W$.

Proof: To prove the upper bound on a set $W \subset \mathbb{R}^{2n}$, we need to show that for each $0 \leq k \leq n-1$

$$\sum_{j=1}^{\infty} (a_j^k(w))^2$$

converges and is bounded on W. Adjusting for the choice of scaling factors, Lemma 14 tells us that the a_j^k's can be written as

$$a_j^k(w) = \gamma_j \sum_{t=0}^{k} \mathcal{A}_{n-k,t} \sum_{\substack{|\alpha|=j-k \\ n_0(\alpha)=k-t}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2\,,$$

where importantly, the $\mathcal{A}_{r,t}$'s don't depend on j. Set

$$A = \max_{r,t} |\mathcal{A}_{r,t}|\,.$$

Then if $|b_1|, ...|b_n| < s$ for some $s > 0$,

$$(a_j^k(w))^2 \leq \gamma_j^2 \left(\sum_{t=0}^{k} A \sum_{\substack{|\alpha|=j-k \\ n_0(\alpha)=k-t}} (b_1^{\alpha_1} ... b_n^{\alpha_n})^2 \right)^2$$

$$\leq \gamma_j^2 \left(\sum_{t=0}^{k} A \sum_{\substack{|\alpha|=j-k \\ n_0(\alpha)=k-t}} s^{2(j-k)} \right)^2 .$$

Now, the number of terms in the sum $\sum_{\substack{|\alpha|=j-k \\ n_0(\alpha)=k-t}} s^{2(j-k)}$ is less than the number of monomials in n variables of degree $j-k$, which is $\frac{(n+j-k-1)!}{(j-k)!(n-1)!}$, so

$$\begin{aligned}\left(a_j^k(w)\right)^2 &\leq \gamma_j^2 k^2 A^2 \left(\frac{(n+j-k-1)!}{(j-k)!(n-1)!} \right)^2 s^{4(j-k)} \\ &=: T_j^k(s) .\end{aligned}$$

We now apply the ratio test to the series $\sum_{j=0}^{\infty} T_j^k(s)$ to check for which values of s it converges. Previously we saw that

$$\lim_{j\to\infty} \frac{\gamma_{j+1}^2}{\gamma_j^2} = \frac{1}{4\pi^2} .$$

Meanwhile

$$\begin{aligned}\lim_{j\to\infty} \left(\frac{(n+j-k)!}{(j+1-k)!(n-1)!} \right)^2 \left(\frac{(j-k)!(n-1)!}{(n+j-k-1)!} \right)^2 &= \lim_{j\to\infty} \left(\frac{(n+j-k)}{(j+1-k)} \right)^2 \\ &= 1 ,\end{aligned}$$

and hence

$$\lim_{j\to\infty} \frac{T_{j+1}^k(s)}{T_j^k(s)} = \frac{s^4}{4\pi^2} .$$

Combining the ratio test with the Weierstrass M test tells us that the series $\sum_{j=0}^{\infty} a_j^k(w)^2$ converges and is bounded provided $|b_1|, ...|b_n| < s$ for some fixed $s < \sqrt{2\pi}$. Combining this with the result on the square summability of the $e_j(x)$'s (Lemma 12) proves the upper bound. □

4.5 Non-zero true parameters

The previous upper bounds for the KL divergence apply when the true function f_T equals zero (or equivalently, the true parameter can be written as $w_0 = 0$). However, most learning problems involve true functions which are non-zero and it is important to understand their KL divergence. Fortunately, the methods we have developed allow us to gain some insight into such models. Specifically, they let us find polynomial bounds for the KL divergence for models where the true function is given by a neural network with a non-zero true parameter.

Starting with a parameterised model whose regression function can be written in the form from the Replacement Strategy,

$$f(x, w) = \sum_{j=0}^{\infty} f_j(w) e_j(x) ,$$

if the true function is given by $f(x, w_0)$ for some true parameter w_0, the difference between the model and true function can be written

$$f(x, w) - f(x, w_0) = \sum_{j=0}^{\infty} (f_j(w) - f_j(w_0)) e_j(x) .$$

There are two methods we could use for finding a polynomial bound for this KL divergence:

Method 1: Repeat the process from Sections 4.1 - 4.4. That is, find a finite basis for the ideal generated by $\{f_j(w) - f_j(w_0)\}_{j=0}^{\infty}$ and check convergence conditions (C2)-(C4). The downside of this method is that it only tells us about the specific choice of w_0, and we would need to repeat the whole process if we changed the true parameter. This process can be very involved to complete and isn't guaranteed to succeed, so having to repeat it many times is not ideal.

Method 2: It would be better if we could work in the opposite direction, first finding a basis which works for any choice of true parameter, showing the convergence conditions, and then substituting in the value of the true parameter w_0. The idea is to consider the "extended" function

$$\begin{aligned} F : X \times W \times W &\longrightarrow \mathbb{R} \\ (x, w, w') &\longmapsto f(x, w) - f(x, w') \,. \end{aligned}$$

This function can be written as

$$F(x, w, w') = \sum_{j=0}^{\infty} F_j(w, w') e_j(x) \,,$$

where $F_j(w, w') = f_j(w) - f_j(w')$. If we can find an N so that $\{F_0, ..., F_N\}$ is a basis for the ideal generated by the F_j's in $\mathbb{R}[w, w']$, we can try to find quotients $A_j^k(w, w')$ in $\mathbb{R}[w, w']$ such that $F_j(w, w') = \sum_{k=0}^{N} A_j^k(w, w') F_k(w, w')$. If $\sum_{j=0}^{\infty} A_j^k(w, w')^2$ converges and is bounded on $W \times W$ for each k, it follows from Lemma 9 that there is $C > 0$ such that

$$\frac{1}{2} \int_X (f(x, w) - f(x, w'))^2 q(x) dx \leq C \sum_{j=0}^{N} F_j(w, w')^2 \tag{4.5.1}$$

for all $(w, w') \in W \times W$. Then for any choice of true parameter $w_0 \in W$, (4.5.1) gives us the polynomial bound on $K(w)$

$$K(w) \leq C \sum_{j=1}^{N} (f_j(w) - f_j(w_0))^2 \,, \tag{4.5.2}$$

and this is equivalent to the bound that would be produced in Method 1 due to the following lemma.

Lemma 16. For any fixed choice of true parameter $w_0 \in W$, $\{F_j(w, w_0)\}_{j=1}^{N}$ is a basis for the ideal generated by $\{f_j(w) - f_j(w_0)\}_{j=0}^{\infty}$ in $\mathbb{R}[w]$.

Proof: We can write

$$F_j(w, w') = \sum_{k=0}^{N} A_j^k(w, w') F_k(w, w')$$

as elements of $\mathbb{R}[w, w']$, where $A_j^k(w, w')$ is also a polynomial in $\mathbb{R}[w, w']$. Then

$$\begin{aligned} f_j(w) - f_j(w_0) =& F_j(w, w_0) \\ =& \sum_{k=0}^{N} A_j^k(w, w_0) F_k(w, w_0) \,. \end{aligned}$$

□

Combining the above with the basis independence of equivalence (Lemma 10) shows that Methods 1 and 2 would provide equivalent upper bounds.

4.5.1 Main theorem: two layer networks with non-zero true parameter

Applying Method 2 to two layer networks with arbitrarily many neurons results in our Main Theorem. Consider the network with n neurons given by

$$f(x, a_1, ..., a_n, b_1, ..., b_n) = \sum_{l=1}^{n} a_l \tanh(b_l x) \,, \tag{4.5.3}$$

which has Taylor series polynomials (up to some constants)

$$f_j(a_1, ..., a_n, b_1, ..., b_n) = \sum_{l=1}^{n} a_l b_l^{2j+1} .$$

Theorem 1 (Main theorem)**.** Given the network f as in (4.5.3) with input space $X = [-t, t]$ for some $0 < t < \frac{\sqrt{\pi}}{2\sqrt{2}}$ and weight space $W = \mathbb{R}^n \times [-s, s]^n$ where $0 < s < \sqrt{2\pi}$, then for any given true parameter $w_0 = (a_{0,1}, ..., a_{0,n}, b_{0,1}, ..., b_{0,n}) \in W$

$$K(w) \leq C \sum_{j=0}^{2n-1} \left(\sum_{l=1}^{n} a_l b_l^{2j+1} - a_{0,l} b_{0,l}^{2j+1} \right)^2 \tag{4.5.4}$$

as functions on W for some constant $C > 0$.

Proof: The extended function for this model is

$$F(x, a_1, ..., b_n, a'_1, ..., b'_n) = f(x, a_1, ...b_n) - f(x, a'_1, ...b'_n) .$$

For convenience write $a = (a_1, ...a_n)$, $b = (b_1, ...b_n)$, $a' = (a'_1, ..., a'_n)$ and $b' = (b_1, ..., b'_n)$. The extended function has Taylor series polynomials

$$F_j(a, b, a', b') = \sum_{l=1}^{n} a_l b_l^{2j+1} - \sum_{k=1}^{n} a'_k b_k'^{2j+1} .$$

First consider the polynomials

$$\tilde{F}_j(a, b, a', b') = \sum_{l=1}^{n} a_l b_l^{2j+1} + \sum_{k=1}^{n} a'_k b_k'^{2j+1} ,$$

which are the Taylor series polynomials for a network with $2n$ hidden neurons. From Section 4.4, $\{\tilde{F}_0, ..., \tilde{F}_{2n-1}\}$ is a basis for the ideal generated by the $\tilde{F}_j$'s. By Lemma 15, if the input space is $X = [-t, t]$ for some $0 < t < \frac{\sqrt{\pi}}{2\sqrt{2}}$ and the input distribution $q(x)$ is uniform, then

$$\frac{1}{2} \int_X \left(\sum_{l=1}^{n} a_l \tanh(b_l x) + \sum_{k=1}^{n} a'_k \tanh(b'_k x) \right)^2 q(x) dx \leq C \sum_{j=1}^{2n-1} \tilde{F}_j(a, b, a', b')^2$$

as functions on the weight space $\mathbb{R}^n \times [-s, s]^n \times \mathbb{R}^n \times [-s, s]^n$ for any $0 < s < \sqrt{2\pi}$. Because $\{\tilde{F}_0, ..., \tilde{F}_{2n-1}\}$ is a basis for the ideal generated by all the $\tilde{F}_j$'s and $F_j(a, b, a', b') = \tilde{F}_j(a, b, -a', b')$, it follows that $\{F_0, F_1, ..., F_{2n-1}\}$ is a basis for the ideal generated by the F_j's and that

$$\frac{1}{2} \int_X \left(\sum_{l=1}^{n} a_l \tanh(b_l x) - \sum_{k=1}^{n} a'_k \tanh(b'_k x) \right)^2 q(x) dx \leq C \sum_{j=0}^{2n-1} F_j(a, b, a', b')^2 \tag{4.5.5}$$

as functions on $\mathbb{R}^n \times [-s, s]^n \times \mathbb{R}^n \times [-s, s]^n$. □

While equivalence requires both lower and upper bounds, as discussed in Lemma 3, upper bounds on the KL divergence can provide upper bounds on the RLCT, and this in turn can provide an upper bound on the generalisation error for a model.

Remark. This result backs up statements made by Aoyagi and Watanabe in their paper *The Zeta Function for Learning Theory and Resolution of Singularities* [AW09]. In this paper they replace $K(w)$ with the same polynomial in the Main Theorem to calculate the RLCT. In justifying this step, they cite a lemma in an earlier paper [Wat99b, Lemma 5], however we have not yet been able to reconstruct a proof of equivalence from that lemma.

The proof of Theorem 1 also pays close attention to convergence of each of the series involved, producing a domain where the polynomial upper bound is guaranteed to hold.

Chapter 5

Zero sets

One of the key points of Singular Learning Theory is that there is a "fundamental relation between algebraic geometry and statistical learning theory" [Wat09, p. vii]. In particular, the geometry of a model's KL divergence determines its generalisation performance. In this Chapter we will illustrate the geometry of the set of true parameters

$$W_0 = \{w \in W \mid K(w) = 0\}$$

in some simple examples.

Studying this geometry can be greatly simplified using an equivalent polynomial, because such a polynomial would automatically have the same zero set as $K(w)$. However, we have not proven both the lower bound and upper bound parts of equivalence: we were only able to prove the upper bound in Chapter 4. Despite this, the process we used to find these upper bounds is sufficient to show they have the same zero sets. In Section 2.4, we saw that if a model satisfies the Replacement Strategy's conditions, then $K(w)$ is zero if and only if all the model's Taylor series polynomials $\{f_j(w)\}_{j=0}^{\infty}$ are zero at w. Further, this will occur precisely when each polynomial in a basis for $I = \langle\{f_j(w)\}_{j=0}^{J}\rangle$ equals zero. Since our upper bounds are comprised of these basis polynomials, their zero sets are the same as that of the KL divergence.

In this Chapter, we turn to a simple application of these basis polynomials, using them to find irreducible decompositions for the sets of true parameters for small neural networks. These decompositions make it easy to plot the sets, and doing so will illustrate interesting phenomena where the geometry of the zero set changes drastically as the true function changes. This behaviour will be reflected in the models' RLCTs, which we will compute using a formula from Aoyagi and Watanabe [AW09].

This Chapter also be helps to develop an intuitive picture of the types of singularities appearing in singular models. Currently the literature contains only sketches of general singularities which appear in algebraic geometry, and these don't accurately represent the zero sets of neural networks.

To find the zero sets, we will start by studying the smallest and simplest models, before slowly increasing their size. First, we look at the zero sets for two layer tanh networks with one, two, and three neurons in the case where the true parameter is zero. We then use the results from this case to study the zero sets for one and two neuron networks when the true parameter is non-zero.

5.1 Zero true parameter

5.1.1 Single neuron

For a network with a single neuron, $f(x, a, b) = a\tanh(bx)$, the KL divergence is bounded by the polynomial $f_0(a, b) = ab$. Hence the zero set of the KL divergence is

$$W_0 = \{a, b \mid ab = 0\}$$

$$= Z(a) \cup Z(b)\,,$$

where given an ideal $I \subset \mathbb{R}[w]$, $Z(I) = \{w \in W \mid f(w) = 0\ \forall f \in I\}$. The ideals $\langle a \rangle$ and $\langle b \rangle$ are both prime, so this gives the irreducible decomposition for W_0.

5.1.2 Two neurons

For the tanh network with two neurons $f(x, a, b, c, d) = a\tanh(bx) + c\tanh(dx)$ with zero true parameter, the set of true parameters is given by solutions to the equations

$$ab + cd = 0\,, \tag{5.1.1}$$
$$ab^3 + cd^3 = 0\,. \tag{5.1.2}$$

For the first equation $ab + cd = 0$ to hold, we must have that $ab = -cd$. If we substitute this into the second equation, $ab^3 + cd^3 = 0$, we get $-cdb^2 + cd^3 = 0$, so

$$cd(d + b)(d - b) = 0\,.$$

From this, we have several cases.

Case 1: $c = 0$

In this case, equation (5.1.1) implies that $ab = 0$, so that either

- Case 1(a): $a = 0$ while b and d are free,
- Case 1(b): $b = 0$ while a and d are free.

Case 2: $d = 0$

Again equation (5.1.1) implies that $ab = 0$, so that either

- Case 2(a): $a = 0$ while b and c are free,
- Case 2(b): $b = 0$ while a and c are free.

Case 3: $d - b = 0$

In this case, equation (5.1.1) gives $ab + cb = 0$, so $b(a + c) = 0$. If $b = 0$, automatically then $d = 0$, so this reduces to case 2. If $b \neq 0$, it follows that $a + c = 0$.

Case 4: $b + d = 0$

Equation (5.1.1) becomes $ab - cb = 0$, so either $b = 0$ reducing to case 1, or $a - c = 0$.

From these four cases we conclude that the zero set for a two neuron network's KL divergence is

$$\begin{aligned} W_0 = & Z(a, c) \cup Z(b, c) && \text{(case 1)} \\ & \cup Z(a, d) \cup Z(b, d) && \text{(case 2)} \\ & \cup Z(b - d, a + c) && \text{(case 3)} \\ & \cup Z(b + d, a - c)\,. && \text{(case 4)} \end{aligned}$$

Each of the above sets is clearly a variety, and none of them are contained in any other, so this gives the irreducible decomposition of W_0.

Each of the above irreducible components is a plane in $\mathbb{R}^4$. We can visualise these planes in 3 dimensions by sketching their image under the projection onto the first three variables $(a, b, c, d) \mapsto (a, b, c)$. This projection is sketched in Figure 5.1.

The zero set is a union of planes in $\mathbb{R}^4$ which all intersect at the origin. The singularities of this set are the intersections between these planes.

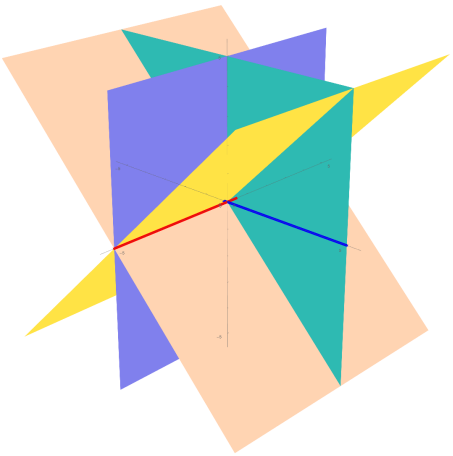


Figure 5.1: The zero set for the KL divergence of a two neuron tanh network when the true parameter is zero. The zero set in $\mathbb{R}^4$ has been projected into $\mathbb{R}^3$. Each plane represents a different irreducible component of the zero set. The red and blue lines are seperate irreducible components, and also represent planes in $\mathbb{R}^4$ but appear as lines when projected into $\mathbb{R}^3$.

5.1.3 Three Neurons

In this Section, we find the irreducible decomposition for the zero set of a three neuron network. Doing so will help us study the zero set for two neuron networks when the true function is given by a single neuron.

For the three neuron network $f(x, a_1, b_1, a_2, b_2, a_3, b_3) = \sum_{i=1}^{3} a_i \tanh(b_i x)$, the zero set of the KL divergence is given by the solutions of the following three equations

$$f_0(a_1, b_1, a_2, b_2, a_3, b_3) = a_1 b_1 + a_2 b_2 + a_3 b_3 = 0\,, \tag{5.1.3}$$

$$f_1(a_1, b_1, a_2, b_2, a_3, b_3) = a_1 b_1^3 + a_2 b_2^3 + a_3 b_3^3 = 0\,, \tag{5.1.4}$$

$$f_2(a_1, b_1, a_2, b_2, a_3, b_3) = a_1 b_1^5 + a_2 b_2^5 + a_3 b_3^5 = 0\,. \tag{5.1.5}$$

In these three equations, it is harder to notice any obvious substitutions or factorisations that make it easy to find the zero set, partly because each of the generators f_0, f_1 and f_2 contain all six of the variables. Instead of working with these polynomials directly, we will find polynomials in the ideal $I = \langle f_0, f_1, f_2 \rangle$ which contain fewer variables, and use these to find the structure of the zero set.

Since f_0, f_1, and f_2 generate the ideal I, they are all zero at a point w if and only if $g(w) = 0$ for all $g \in I$. If we can find a $g \in I$ which involves for example only two or three of the variables, we can find conditions on these variables which must hold for f_0, f_1, and f_2 to be zero. For example, we could find that for some $g \in I$ to be zero, it must be true that $b_1 = b_2$. It would follow that for f_0, f_1, and f_2 to be zero, b_1 must equal b_2, and we could use this substitution to simplify the three polynmials.

To find elements of the ideal I which contain fewer variables, we use the Elimination Theorem.

Theorem. *(Elimination Theorem)* Suppose $I \subset \mathbb{R}[x_1, ..., x_n]$ is an ideal and G is a Gröbner basis for I using lexicographic order, then for each $0 \leq l < n$

$$G \cap \mathbb{R}[x_{l+1}, ..., x_n]$$

is a Gröbner basis for the ideal $I \cap \mathbb{R}[x_{l+1}, ..., x_n]$.

A proof of this theorem appears in Theorem 2 of Chapter 3 in [CLO15].

The Elimination Theorem lets us find elements of the ideal I which contain fewer of the variables. Suppose we wish to find every polynomial in I involving only the variables $x_{i_1}, ..., x_{i,r}$ $(r < n)$. If we use a modified lexicographic order, where $x_{i_1}, ..., x_{i,r}$ are smaller than the other variables, finding a Gröbner basis for I with this order will provide all such polynomials.

Remark. Washino and Takahashi use this theorem to study zero sets of neural networks, using it to find a parameterisation for the zero set of a two layer tanh network [WT21].

Using Singular, a Gröbner basis with standard lexicographic order for the ideal $\langle f_0, f_1, f_2 \rangle$ has nine polynomials. We don't need all nine of them though, as the first one is enough for us to progress with the calculation of the zero set. The first polynomial in this Gröbner basis is

$$g_1 = a_2 b_2^5 a_3 b_3 - 2a_2 b_2^3 a_3 b_3^3 + a_2 b_2 a_3 b_3^5 \,.$$

This factors as

$$g_1 = a_2 b_2 a_3 b_3 (b_2^2 - b_3^2)^2 \,,$$

so for f_0, f_1 and f_2 to all be zero, it must be true that either one of a_2, b_2, a_3, and b_3 is zero, or $b_2^2 = b_3^2$.

In the case where $a_i = 0$ or $b_i = 0$ (where $i \in \{1, 2, 3\}$), f_1, f_2 and f_3 become

$$f_j = \sum_{\substack{l=1,2,3 \\ l \neq i}} a_l b_l^{2j+1} \,,$$

which equal the Taylor series polynomials of a two neuron network with $a_r \tanh(b_r x)$ and $a_t \tanh(b_t x)$ as the neurons, where we write $\{r, t\}$ for $\{1, 2, 3\} \backslash \{i\}$. When this occurs, we know by Lemma 14 that $f_2 \in \langle f_0, f_1 \rangle$.

Case 0: $a_i = 0$ or $b_i = 0$ $(i \in \{1, 2, 3\})$

Using the results of the two neuron example in the previous section, when $a_i = 0$ or $b_i = 0$, f_0, f_1, and f_2 are all zero if and only if

- $a_r = 0$ and $a_t = 0$, or
- $b_r = 0$ and $a_t = 0$, or
- $a_r = 0$ and $b_t = 0$, or
- $b_r = 0$ and $b_t = 0$, or
- $a_r + a_t = 0$ and $b_r - b_t = 0$, or
- $a_r - a_t = 0$ and $b_r + b_t = 0$.

If none of a_1, a_2, a_3, b_1, b_2, b_3 are zero, the equation for g_1 tells us that we must have $b_3 = \pm b_2$.

Case 1: $b_3 = b_2$

With this substitution, the Taylor series polynomials become $f_0' = a_1 b_1 + a_2 b_2 + a_3 b_2$, $f_1' = a_1 b_1^3 + a_2 b_2^3 + a_3 b_2^3$, and $f_3' = a_1 b_1^5 + a_2 b_2^5 + a_3 b_2^5$. Using Singular, a Gröbner basis for their ideal with lexicographic order is

$$\begin{aligned} h_1 &= b_1^2 a_2 b_2 + b_1^2 b_2 a_3 - a_2 b_2^3 - b_2^3 a_3 \,, \\ h_2 &= a_1 a_2 b_2^3 + a_1 b_2^3 a_3 + b_1 a_2^2 b_2^2 + 2 b_1 a_2 b_2^2 a_3 + b_1 b_2^2 a_3^2 \,, \\ h_3 &= a_1 b_1 + a_2 b_2 + b_2 a_3 \,, \end{aligned}$$

but it's not clear that these can be factored in a way which helps to calculate the zero sets. Instead, if we use lexicographic order but with the remaining variables ordered by $b_2 > a_3 > a_2 > b_1 > a_1$, a Gröbner basis for $\langle f_0, f_1, f_2 \rangle$ is

$$p_1 = a_3^2 b_1^3 a_1 + 2 a_3 a_2 b_1^3 a_1 + a_2^2 b_1^3 a_1 - b_1^3 a_1^3 \,,$$

$$
\begin{aligned}
p_2 &= b_2 b_1^2 a_1^2 + a_3 b_1^3 a_1 + a_2 b_1^3 a_1 \,, \\
p_3 &= b_2 a_3 + b_2 a_2 + b_1 a_1 \,, \\
p_4 &= b_2^2 b_1 a_1 - b_1^3 a_1 \,.
\end{aligned}
$$

The polynomial p_4 then factors as $p_4 = b_1 a_1 (b_2^2 - b_1^2)$, so for p_4 to equal 0, either b_1 or a_1 must be zero or b_2 must equal $\pm b_1$. If $b_1 = 0$ or $a_1 = 0$, we are brought back to the situation in Case 0.

Case 1a: $b_2 = b_1$

In this case, the polynomials become $f_0'' = a_1 b_1 + a_2 b_1 + a_3 b_1$, $f_1'' = a_1 b_1^3 + a_2 b_1^3 + a_3 b_1^3$ and $f_2'' = a_1 b_1^5 + a_2 b_1^5 + a_3 b_1^5$. These are all zero when either $b_1 = 0$ (which has already been covered by Case 0) or $a_1 + a_2 + a_3 = 0$.

Case 1b: $b_2 = -b_1$

The polynomials become $f_0''' = a_1 b_1 - a_2 b_1 - a_3 b_1$, $f_1''' = a_1 b_1^3 - a_2 b_1^3 - a_3 b_1^3$ and $f_2''' = a_1 b_1^5 - a_2 b_1^5 - a_3 b_1^5$, which are zero when either $b_1 = 0$ (already covered by Case 0) or $a_1 - a_2 - a_3 = 0$.

Case 2: $b_3 = -b_2$

We can use Case 1 to find the zero sets when $b_3 = -b_2$. Plugging $b_3 = -b_2$ into f_0, f_1, and f_2 results in the polynomials $\tilde{f}_0 = a_1 b_1 + a_2 b_2 - a_3 b_2$, $\tilde{f}_1 = a_1 b_1^3 + a_2 b_2^3 - a_3 b_2^3$, and $\tilde{f}_2 = a_1 b_1^5 + a_2 b_2^5 - a_3 b_3^5$. Now, $\tilde{f}_i(a_1, b_1, a_2, b_2, a_3) = f_i'(a_1, b_1, a_2, b_2, -a_3)$ for $i = 0, 1, 2$. This implies that $\tilde{f}_0$, $\tilde{f}_1$ and $\tilde{f}_2$ are all zero at a point $(a_1, b_1, a_2, b_2, a_3)$ if and only if $(a_1, b_1, a_2, b_2, -a_3)$ lies in the zero set of f_0', f_1' and f_2'.

Case 2a: $b_2 = b_1$ (as in case 1a)

In this case, the polynomials are zero when $b_1 = 0$ or $a_1 + a_2 - a_3 = 0$.

Case 2b: $b_2 = -b_1$ (as in case 1b)

In this case, the polynomials are zero when $b_1 = 0$ or $a_1 - a_2 + a_3 = 0$.

Putting together the results of each case, we find that the zero set for the KL divergence of a 3 neuron network is

$$
\begin{aligned}
W_3 = & Z(a_1, a_2, a_3) \cup Z(a_1, a_3, b_2) \cup Z(a_1, a_2, b_3) \cup Z(a_1, b_2, b_3) \\
& \cup Z(a_2, b_1, a_3) \cup (a_2, b_1, b_3) \cup Z(a_3, b_1, b_2) \cup Z(b_1, b_2, b_3) \\
& \cup Z(a_1, a_2 + a_3, b_2 - b_3) \cup Z(a_1, a_2 - a_3, b_2 + b3) \\
& \cup Z(a_2, a_1 + a_3, b_1 - b_3) \cup Z(a_2, a_1 - a_3, b_1 + b_3) \\
& \cup Z(a_3, a_1 + a_2, b_1 - b_2) \cup Z(a_3, a_1 - a_2, b_1 + b_2) \\
& \cup Z(b_1, a_2 + a_3, b_2 - b_3) \cup Z(b_1, a_2 - a_3, b_2 + b3) \\
& \cup Z(b_2, a_1 + a_3, b_1 - b_3) \cup Z(b_2, a_1 - a_3, b_1 + b_3) \\
& \cup Z(b_3, a_1 + a_2, b_1 - b_2) \cup Z(b_3, a_1 - a_2, b_1 + b_2) \\
& \cup Z(b_2 - b_3, b_1 - b_2, a_1 + a_2 + a_3) \\
& \cup Z(b_2 - b_3, b_1 + b_2, a_1 - a_2 - a_3) \\
& \cup Z(b_2 + b_3, b_1 - b_2, a_1 + a_2 - a_3) \\
& \cup Z(b_2 + b_3, b_1 + b_2, a_1 - a_2 + a_3) \,,
\end{aligned} \tag{5.1.6}
$$

which is a union of many 3 dimensional subspaces of $\mathbb{R}^6$.

Remark. By writing the tanh's in a network in terms of exponentials, and using their linear independence, it's possible to find the same decomposition of W_3 without using the equivalent polynomials. However, this method seems like it would only work for two layer networks, and not generalise well to other models.

5.2 Non-zero true parameters and RLCTs

Having found the irreducible decompositions of the zero sets for single neuron, two neuron, and three neuron networks, we can use the results of Section 5.1 to study this decomposition for networks where the true function is non-zero. This will reveal surprising properties of the structure of the zero set as we change the value of the true parameter.

When searching for polynomial upper bounds in Chapter 4, the main technique for studying the case with a non-zero true parameter was to invent a network with extra neurons, find a polynomial upper bound for the KL divergence for this larger network, and then substitute the true parameters into this polynomial. A similar approach works here, where we can use the irreducible decompositions in the last section to find decompositions for the case with non-zero true parameters. More specifically, we can study the zero sets for a network with n neurons and true function with m neurons ($m \leq n$) by using the irreducible decomposition of the zero set for a network with $n+m$ neurons and a zero true function.

5.2.1 Single neuron model with one true neuron

Consider the single neuron network from Section 5.1.1, $f(x,a,b) = a\tanh(bx)$, but now with a true function given by a single neuron network

$$f_T(x) = a_T \tanh(b_T x)\,,$$

where $w_T = (a_T, b_T)$ is some fixed true parameter. We write $W_{1,1} \subset \mathbb{R}^2$ for the zero set for this model and true parameter.

The KL divergence for this model, $K(w)$, is zero if and only if the polynomials $ab - a_T b_T$ and $ab^3 - a_T b_T^3$ are both zero. This occurs whenever $f_0(a,b,-a_T,b_T) = f_1(a,b,-a_T,b_T) = 0$, where

$$\begin{aligned} f_0(a,b,c,d) &= ab + cd\,, \\ f_1(a,b,c,d) &= ab^3 + cd^3\,, \end{aligned}$$

are the polynomials from the two neuron network with zero true parameter. Writing $W_2 \subset \mathbb{R}^4$ for the zero sets of these polynomials, in Section 5.1.2 we saw that

$$\begin{aligned} W_2 =& Z(a,c) \cup Z(b,c) \cup Z(a,d) \cup Z(b,d) \\ & \cup Z(b-d, a+c) \cup Z(b+d, a-c)\,. \end{aligned}$$

Hence, for the single neuron network with a single true neuron, $K(w) = 0$ at a point $w = (a,b)$ if and only if $(a,b,-a_T,b_T) \in W_2$. This occurs only when

$$\begin{aligned} (a,b) \;\in\; & Z(a,-a_T) \cup Z(b,-a_T) \cup Z(a,b_T) \cup Z(b,b_T) \\ & Z(b-b_T, a-a_T) \cup Z(b+b_T, a+a_T)\,. \end{aligned} \tag{5.2.1}$$

In the above notation we treat a and b as variables, and a_T and b_T as fixed real numbers. We define the set $Z(a,-a_T)$ to be $Z(a) \subset \mathbb{R}^2$ if $a_T = 0$, and define it to be empty if $a_T \neq 0$. We define the sets $Z(b,-a_T)$ and $Z(a,b_T)$ similarly.

The expression in (5.2.1) shows that the structure of the zero set for this model and true function, $W_{1,1}$, is strongly influenced by the value of the true parameter (a_T, b_T). For example if $a_T \neq 0$ the set $Z(a,-a_T)$ is empty, while if a_T is zero, this set is the line along the b axis. We can see all the possible forms of $W_{1,1}$ using the following cases.

Case 1: $a_T = 0$ or $b_T = 0$

In this case the true function f_T is zero, so the zero set is just that of a single neuron network with zero true parameter.

$$W_{1,1} = Z(a) \cup Z(b)\,,$$

which is two lines along the axes, intersecting at the origin.

Case 2: $a_T \neq 0$ and $b_T \neq 0$

In this case, the sets $Z(a, -a_T)$, $Z(b, -a_T)$, $Z(a, b_T)$, and $Z(b, b_T)$ are all empty. Meanwhile, the set $Z(b - b_T, a - a_T)$ is just the point (a_T, b_T), and $Z(b + b_T, a + a_T)$ is the point $(-a_T, -b_T)$.

In this case $W_{1,1}$ is just a pair of points in $\mathbb{R}^2$

$$W_{1,1} = \{(a_T, b_T), (-a_T, -b_T)\}\,.$$

The way $W_{1,1}$ changes as either true parameter approaches zero is quite interesting. If $b_T \to 0$, the two points in the zero set (a_T, b_T) and $(-a_T, -b_T)$, approach the a axis. As soon as they reach the axis though, the whole of the a and b axes become the zero set. Figure 5.2 illustrates this change in the structure of $W_{1,1}$.

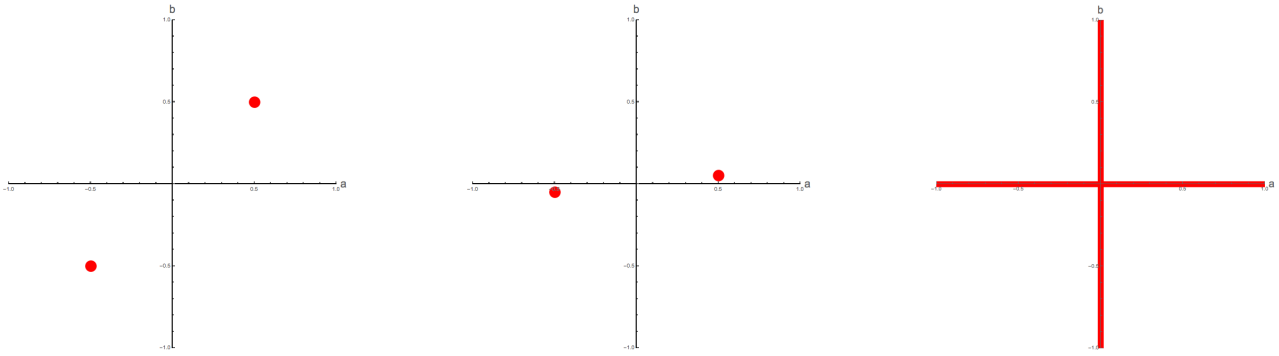
At first it's surprising that this set suddenly changes from a pair of points to an unbounded set. It is hard to picture how the KL divergence must behave for this to happen. But this becomes clearer by looking at the integral which defines $K(w)$. The KL divergence is given by the integral

$$K(w) = \int q(x)\left(a\tanh(bx) - a_T\tanh(b_T x)\right)^2 dx\,.$$

When either a or b equal 0, this reduces to

$$K(w) = \int q(x) a_T^2 \tanh^2(b_T x) dx\,,$$

which is a non-negative constant. The axes are therefore always a level set of $K(w)$, regardless of the value of the true parameter (a_T, b_T). As a_T or b_T approach zero, this level set lowers as a whole and approaches zero.



(a) (b) (c)

Figure 5.2: The set of true parameters $W_{1,1}$ for the single neuron network with single true neuron are shown in (a), (b), and (c) for different values of the true parameter (a_T, b_T). (a) shows $W_{1,1}$ when $a_T, b_T = 0.5$, (b) shows the set when $a_T = 0.5$ and $b_T = 0.05$, and (c) shows the set when $b_T = 0$.

Figure 5.2 shows clearly how the geometry of $K(w)$ changes as the true parameter (a_T, b_T) changes. Using Aoyagi and Watanabe's formula for the RLCT of a two layer model [AW09], we can see how the RLCT (and hence the generalisation error) of a two layer model changes with this geometry.

The formula which Aoyagi and Watanabe found for the RLCT of two layer networks with non-zero true parameters is quite complicated. The value of the RLCT splits along many cases, based on how many of the true parameter's components are zero or equal another of its components. In Section

5.2.2, we will discuss how the formula works in general. However, for a single neuron network with true function also given by a single neuron, the RLCT is simple to describe.

For this model and true function, the RLCT λ, is

$$\lambda = \begin{cases} 1, & \text{if } (a_T, b_T) \neq (0,0) \\ \frac{1}{2}, & \text{if } a_T = 0 \text{ or } b_T = 0 \end{cases} . \tag{5.2.2}$$

In situations like Figure 5.2a, and Figure 5.2b, where the set of true parameters is just a pair of points, the RLCT is larger than when the set of true parameters is the a and b axes, as in Figure 5.2c.

5.2.2 Two neuron model with one true neuron

Having observed interesting behaviour in zero sets for a single neuron, we will perform a similar analysis for a network with two neurons.

The largest network we were able to analyse in the previous section had three neurons. Hence we're able to study a two neuron network with a single true neuron. Consider the two neuron network from Section 5.1.2, but now with a true function given by $f_T(x) = a_T \tanh(b_T x)$, where a_T and $b_T \in \mathbb{R}$ are a fixed choice of true parameter.

Denoting the zero set for the KL divergence of this model and true function by $W_{2,1}$, the Taylor series implies that this set equals the zero set of the following three polynomials

$$\begin{aligned} f_0(a,b,c,d) &= ab + cd - a_T b_T \,, \\ f_1(a,b,c,d) &= ab^3 + cd^3 - a_T b_T^3 \,, \\ f_2(a,b,c,d) &= ab^5 + cd^5 - a_T b_T^5 \,. \end{aligned}$$

We can write these polynomials as $f_j(a,b,c,d) = F_j(a,b,c,d,-a_T,b_T)$ for $j = 0,1,2$, where

$$F_j(a_1,b_1,a_2,b_2,a_3,b_3) = a_1 b_1^{2j+1} + a_2 b_2^{2j+1} + a_3 b_3^{2j+1}$$

are the first three Taylor series polynomials for a three neuron network. Hence

$$\begin{aligned} (a,b,c,d) \in W_{2,1} &\iff F_j(a,b,c,d,-a_T,b_T) = 0, \qquad j = 0,1,2 \\ &\iff (a,b,c,d,-a_T,b_T) \in W_3 \,, \end{aligned} \tag{5.2.3}$$

where $W_3 \subset \mathbb{R}^6$ is the zero set for the three polynomials F_0, F_1 and F_2. The irreducible decomposition of W_3 is written in equation (5.1.6). Using this decomposition, (5.2.3) implies that

$$\begin{aligned} W_{2,1} = &Z(a,c,-a_T) \cup Z(a,-a_T,d) \cup Z(a,c,b_T) \cup Z(a,d,b_T) \\ &\cup Z(c,b,-a_T) \cup (c,b,b_T) \cup Z(-a_T,b,d) \cup Z(b,d,b_T) \\ &\cup Z(a,c-a_T,d-b_T) \cup Z(a,c+a_T,d+b_T) \\ &\cup Z(c,a-a_T,b-b_T) \cup Z(c,a+a_T,b+b_T) \\ &\cup Z(-a_T,a+c,b-d) \cup Z(-a_T,a-c,b+d) \\ &\cup Z(b,c-a_T,d-b_T) \cup Z(b,c+a_T,d+b_T) \\ &\cup Z(d,a-a_T,b-b_T) \cup Z(d,a+a_T,b+b_T) \\ &\cup Z(b_T,a+c,b-d) \cup Z(b_T,a-c,b+d) \\ &\cup Z(d-b_T,b-d,a+c-a_T) \\ &\cup Z(d-b_T,b+d,a-c+a_T) \\ &\cup Z(d+b_T,b-d,a+c+a_T) \\ &\cup Z(d+b_T,b+d,a-c-a_T) \,. \end{aligned} \tag{5.2.4}$$

In the above notation, the variables a, b, c, and d are treated as functions, while a_T and b_T are fixed real numbers. We interpret a set like $Z(a, c, a_T)$ to equal $Z(a, c) \subset \mathbb{R}^4$ if $a_T = 0$, and define it to be the empty set if $a_T \neq 0$.

As in the single neuron example there are two cases based on the value of the true parameter. In each case we can just read off the irreducible decomposition of $W_{2,1}$ from the above equation.

Case 1: $a_T = 0$ or $b_T = 0$

In this case

$$\begin{aligned} W_{2,1} =& Z(a, c) \cup Z(a, d) \cup Z(c, b) \cup Z(b, d) \\ & \cup Z(a + c, b - d) \cup Z(a - c, b + d)\,. \end{aligned}$$

Several of the components disappeared when setting a_T or b_T to zero because they become subsets of other components. For example $Z(d - b_T, b - d, a + c - a_T)$ equals $Z(d - b_T, b - d, a + c)$ when $a_T = 0$, which is a subset of $Z(a + c, b - d)$. Likewise, when $b_T = 0$, $Z(d - b_T, b - d, a + c)$ equals $Z(d, b - d, a + c)$ which is also contained in $Z(b - d, a + c)$.

Note that in this case $W_{2,1}$ equals the zero set from the two neuron network with zero true function that we found in Section 5.1.2. This is to be expected, as it is the exact same model and true function. In our sketch we saw that this is a union of 6 planes in $\mathbb{R}^4$ which all intersect at the origin.

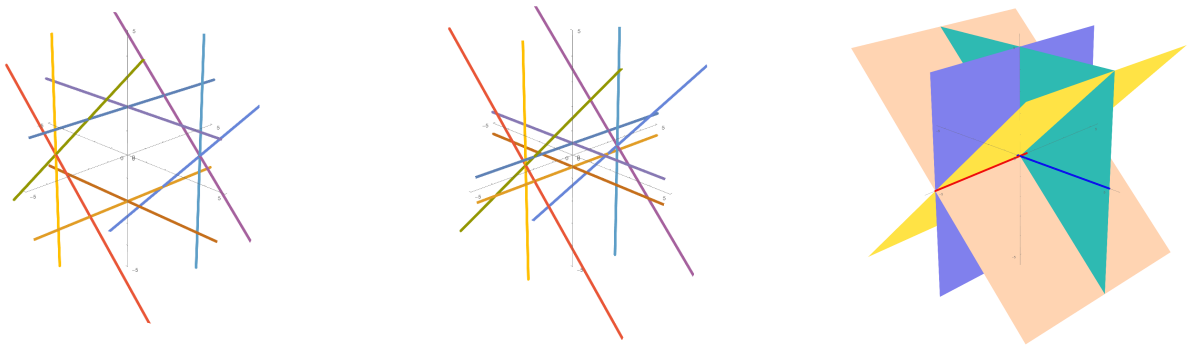
Case 2: $a_T \neq 0$ and $b_T \neq 0$

In this case several of the sets in (5.2.4) are empty, leaving

$$\begin{aligned} W_{2,1} =& Z(a, c - a_T, d - b_T) \cup Z(a, c + a_T, d + b_T) \\ & \cup Z(c, a - a_T, b - b_T) \cup Z(c, a + a_T, b + b_T) \\ & \cup Z(b, c - a_T, d - b_T) \cup Z(b, c + a_T, d + b_T) \\ & \cup Z(d, a - a_T, b - b_T) \cup Z(d, a + a_T, b + b_T) \\ & \cup Z(d - b_T, b - d, a + c - a_T) \\ & \cup Z(d - b_T, b + d, a - c + a_T) \\ & \cup Z(d + b_T, b - d, a + c + a_T) \\ & \cup Z(d + b_T, b + d, a - c - a_T)\,. \end{aligned}$$

Each set in the above decomposition is a line in $\mathbb{R}^4$, rather than a plane as seen in Case 1.

Using this decomposition, we can plot the image of $W_{2,1}$ under the projection $(a, b, c, d) \mapsto (a, b, c)$ in $\mathbb{R}^3$, which is shown for different values of the true parameter (a_T, b_T) in Figure 5.3.



(a) (b) (c)

Figure 5.3: The zero sets for a two neuron network with true function given by a single neuron with different values of the true parameter. (a) shows the zero set with $a_T = b_T = 2$, while in (b) $a_T = 0.5$, $b_T = 2$, and in (c) $a_T = 0$. Figure (c) is the same zero set as in Figure 5.1

The zero set behaves similarly to the single neuron case when we vary the true parameter. When the true parameter is non-zero, $W_{2,1}$ is a collection of 1D lines in $\mathbb{R}^4$, but as soon as either component of the true parameter equals zero, $W_{2,1}$ turns into a collection of 2D planes.

This raises the question of whether the zero set for the two neuron network with zero true parameter, W_2, is always a level set of the KL divergence for the two neuron network even when the true parameter is non-zero. This is true as a consequence of the following general lemma.

Lemma 17. If a model is given by a parametric function $f(x, w)$, and the true distribution is given by some true function $f_T(x)$, then the KL divergence $K(w)$ is constant on the set

$$W' = \{w \in W \mid f(x, w) = 0 \ \forall x \in \text{supp}\{q(x)\}\}\,.$$

Proof: This lemma follows imediately from writing $K(w) = \int_X (f(x, w) - f_T(x))^2 \, q(x)dx$.

Formula for the RLCT

In Section 5.2.1 we used Aoyagi and Watanabe's formula, which is outlined in [AW09], to compute the RLCT of a single neuron network with true function given by a single neuron. Here we apply the same formula to a network with two neurons and a single true neuron, but first we will explain how the formula works for a general network with n neurons.

Consider the n neuron network $f(x, a_1, b_1, ..., a_n, b_n) = \sum_{i=1}^{n} a_i \tanh(b_i x)$, with true function $f_T(x) = f(x, a^*_{T,1}, b^*_{T,1}, ..., a^*_{T,n}, b^*_{T,n})$, where $w^*_T = (a^*_{T,1}, b^*_{T,1}, ..., a^*_{T,n}, b^*_{T,n})$ is a choice of true parameter.

Let

$$W_0 = \{w \in W \mid f(x, w) = f(x, w^*_T) \ \forall x \in \mathbb{R}\}$$

be the set of true parameters. The formula for the RLCT works by computing a local RLCT for each $w \in W_0$, and minimising over each of these constants.

Aoyagi and Watanabe derived the formula for the RLCT by replacing $K(w)$ with the polynomial

$$\sum_{j=0}^{P} \left(\sum_{i=1}^{n} a_i b_i^{2j+1} - a_{T,i} b_{T,i}^{2j+1} \right)^2 ,$$

"where P is a sufficiently large integer", and $w_T = (a_{T,1}, b_{T,1}, ..., a_{T,n}, b_{T,n}) \in W_0$ [AW09]. Note that our upper bound in the Main Theorem is precisely this polynomial with $P = n - 1$.

For a given choice of true parameter $w_T = (a_{T,1}, b_{T,1}, ..., a_{T,n}, b_{T,n})$, the first step in calculating the RLCT is to take all of the true weights appearing inside the tanhs, $b_{T,1}, ..., b_{T,n}$, and form a set containing the non-zero ones $B = \{b_{T,i} \mid 1 \leq i \leq n \text{ s.t } b_{T,i} \neq 0\}$. Define an equivalence relation on B by

$$b_{T,i} \sim b_{T,j} \iff b_{T,i}^2 = b_{T,j}^2 \,.$$

Let $\hat{b}_1, ..., \hat{b}_r$ be representatives of distinct equivalence classes of elements in B, so $0 \leq r \leq n$.

For each $1 \leq i \leq r$, define

$$\hat{a}_i = -\frac{1}{\hat{b}_i} \left(\sum_{\{1 \leq m \leq n \mid b_{T,m}^2 = \hat{b}_i^2\}} a_{T,m} b_{T,m} \right) ,$$

and define $\tilde{r}$ to be the number of non-zero $\hat{a}_i$'s. Reordering these variables, we can assume $\hat{a}_1, \hat{a}_2, ..., \hat{a}_{\tilde{r}}$ are non-zero, while $\hat{a}_{\tilde{r}+1}, ..., \hat{a}_r$ all are zero.

Definition 26. We call $\hat{a}_1, \hat{a}_2, ..., \hat{a}_{\tilde{r}}$ and $\hat{b}_1, ..., \hat{b}_r$ *the modified true parameters.*

Aoyagi and Watanabe note that these modified true parameters satisfy

$$\sum_{i=1}^{n} a_{T,i} b_{T,i}^{2j+1} = -\sum_{i=1}^{\tilde{r}} \hat{a}_i \hat{b}_i^{2j+1}$$

for all $j \in \mathbb{N}_{\geq 0}$.

With these modified true parameters, define the set $B_\tau = \{i \mid b_{T,i}^2 = \hat{b}_i^2\}$ for each $1 \leq \tau \leq r$, and record its size as $s_\tau = \#B_\tau$. Also let $B_0 = \{i \mid b_{T,i} = 0\}$ and $s_0 = \#B_0$.

With these definitions, they compute the following integers. These depend on the number of unique components of the true parameter which are non-zero.

$$\begin{aligned}
n_0 &= \max\{i \in \mathbb{Z} \mid i^2 \leq s_0\} \,, \\
n_{\tau_1} &= 1 + \max\{i \in \mathbb{Z} \mid i^2 + i \leq 2s_{\tau_1} \} \quad \text{for } 1 \leq \tau_1 \leq \tilde{r} \,, \\
n_{\tau_2} &= 1 + \max\{i \in \mathbb{Z} \mid i^2 + i \leq 2(s_{\tau_2} - 1)\} \quad \text{for } \tilde{r} < \tau_2 \leq r \,,
\end{aligned}$$

which are used to calculate

$$\begin{aligned}
\lambda_0 &= \frac{n_0^2 + n_0 + s_0}{4n_0 + 2} \,, \\
\lambda_{\tau_1} &= \frac{n_{\tau_1} + n_{\tau_1}^2 + 2s_{\tau_1}}{4n_{\tau_1}} \quad \text{for } 1 \leq \tau_1 \leq \tilde{r} \,, \\
\lambda_{\tau_2} &= \frac{n_{\tau_2} + n_{\tau_2}^2 + 2(s_{\tau_2} - 1)}{4n_{\tau_2}} \quad \text{for } \tilde{r} < \tau_2 \leq r \,.
\end{aligned}$$

They then compute the local RLCT

$$\lambda_{w_T} = \sum_{\tau=0}^{r} \lambda_\tau \,.$$

The final step in computing the RLCT, is to minimise over all the true parameters

$$\lambda = \min_{w_T \in W_0} \{\lambda_{w_T}\} \,.$$

Remark. The formula for the RLCT works by finding the modified true parameters $\hat{a}_i$ for $1 \leq i \leq \tilde{r}$ and $\hat{b}_i$ for $1 \leq i \leq r$ (recall $\tilde{r} \leq r$), and counting several related quantities. Calling $(\hat{a}_1, ..., \hat{a}_{\tilde{r}}, \hat{b}_1, ..., \hat{b}_{\tilde{r}})$

the *cut modified true parameter* (since we've cut out $\hat{b}_{\tilde{r}+1}, ..., \hat{b}_r$), the following equality between the original and cut modified true parameters holds for all $j \in \mathbb{N}_{\geq 0}$

$$\sum_{i=1}^{n} a_{T,i} b_{T,i}^{2j+1} = -\sum_{i=1}^{\tilde{r}} \hat{a}_i \hat{b}_i^{2j+1} .$$

In fact, it's also true that

$$\sum_{i=1}^{n} a_{T,i} \tanh\left(b_{T,i}^{2j+1} x\right) = -\sum_{i=1}^{\tilde{r}} \hat{a}_i \tanh\left(\hat{b}_i x\right)$$

for all $x \in \mathbb{R}$. This says that the cut modified true parameter gives the same function as the original true parameter (with a sign difference).

The cut modified true parameter gives what is known as a *minimal* version of the true network.

Sussman defined minimal networks while studying how different parameters in two layer tanh networks can give rise to the same function [Sus92]. A minimal network is a neural network which is not equal (as a function) to any network with fewer neurons. In the case of biasless two layer tanh networks, $f(x, w) = \sum_{l=1}^{N} c_l \tanh(d_l x)$ is minimal for the specific choice of parameter $w = (c_1, ..., c_N, d_1, ..., d_N)$ if the following three conditions hold

- $c_l \neq 0$ for all l,
- $d_l^2 \neq d_k^2$ for all $l \neq k$,
- $d_l \neq 0$ for all l.

By construction, these conditions hold for the cut modified true parameter. Hence, the first step in using the RLCT formula is to find a minimal version of the true network.

We should note that this minimal network does not alone determine the RLCT, since the parameters which are cut out $(\hat{b}_{\tilde{r}+1}, ..., \hat{b}_r)$ also affect the expressions for the n_{τ_2}'s and λ_{τ_2}'s.

The RLCT for a two neuron network with single true neuron

We can use this formula to calculate the RLCT of the two neuron network where the true function is a network with a single neuron. Since the formula requires the true network to have the same number of neurons as the model, we will view the true function as a two neuron network, where the second neuron outputs zero. That is, the true network is given by

$$f_T(x) = a_{T,1} \tanh(b_{T,1}) + a_{T,2} \tanh(b_{T,2}) ,$$

where we choose $a_{T,2}$ and $b_{T,2}$ so that $a_{T,2} \tanh(b_{T,2} x) = 0$. We can then vary the parameters for the remaining neuron $a_{T,1} \tanh(b_{T,1} x)$ and see how the RLCT changes. There are two cases for the behaviour of this remaining neuron which affect the RLCT.

Case 1: $a_{T,1} \tanh(b_{T,1} x) \neq 0$

For the true function to be non-zero, both of $a_{T,1}$ and $b_{T,1}$ must be non-zero. However, there are several ways that the second neuron $a_{T,2} \tanh(b_{T,2} x)$ can be made to equal zero. In other words, there are several types of points in the set of true parameters, $w_T \in W_0$, and we need to calculate the local RLCT for each type of point. For example $a_{T,2}$ could be zero while $b_{T,2}$ is non-zero and vice-versa, or otherwise both $a_{T,2}$ and $b_{T,2}$ could be zero.

The complete list of cases to check is

- (a) $a_{T,2} = 0$ and $b_{T,2} = 0$,
- (b) $a_{T,2} \neq 0$ and $b_{T,2} = 0$,

- (c) $a_{T,2} = 0$ and $b_{T,2}^2 = b_{T,1}^2$,
- (d) $a_{T,2} = 0$ and $b_{T,2} \neq 0$ and $b_{T,2}^2 \neq b_{T,1}^2$.

In the third case, for example, the set $B = \{1\}$ so $r = 1$. Since $b_{T,2}^2 = b_{T,1}^2$, it follows that $s_1 = 2$. Meanwhile, $\tilde{r} = 1$ and additionally $s_0 = 0$. With these numbers we compute that $n_0 = 0$, $n_1 = 2$ and hence

$$\begin{aligned}\lambda_{w_T} &= \lambda_1 \\ &= \frac{5}{4}.\end{aligned}$$

The following table shows the local RLCT in each of the four cases

Case 1	Case 2	Case 3	Case 4
$\frac{3}{2}$	$\frac{3}{2}$	$\frac{5}{4}$	$\frac{3}{2}$

The minimum is $\frac{5}{4}$ and so the RLCT of the two neuron network with true function given by a single non-zero neuron is

$$\lambda = \frac{5}{4}.$$

Case 2: $a_{T,1}\tanh(b_{T,1}x) = 0$

Here the true function $f_T(x)$ is zero. There are ten types of points in the set of true parameters which we need to calculate local RLCTs for. Each case depends on which of the components $a_{T,1}, b_{T,1}, a_{T,2}, b_{T,2}$ are zero, and which are non-zero, and whether $b_{T,1} = \pm b_{T,2}$.

We'll only include the calculation for the case $w_T = (0, 0, 0, 0)$. In this case, the set B is empty, and $r = 0$. Likewise, $\tilde{r} = 0$. There are no B_τ sets or s_τ values for $\tau \geq 1$, while $B_0 = \{1, 2\}$ and $s_0 = 2$.

Hence, $n_0 = 1$ and n_τ does not exist for $\tau \geq 1$. It follows

$$\begin{aligned}\lambda_{w_T} &= \lambda_0 \\ &= \frac{2}{3}.\end{aligned}$$

The calculation for every other true parameter yields a local RLCT of either $\frac{2}{3}$ or 1. Taking the minimum, the RLCT of the two neuron network with zero true function is

$$\lambda = \frac{2}{3}.$$

As we saw in the single neuron network, the geometry of the zero set changes at the same points as the RLCT. Whenever the true parameter is non-zero, the zero set is a union of lines as in Figure 5.3a and 5.3b, and the RLCT is $\frac{5}{4}$. As soon as the true parameter equals zero though, the zero set turns into the union of planes shown in Figure 5.3c, and the RLCT decreases to $\frac{2}{3}$.

Chapter 6

Conclusion

In this thesis, we have studied how the KL divergence for neural networks can be bounded by a polynomial. This led to Theorem 1, which provides a polynomial bound in the case of two layer tanh networks. Our approach to proving this theorem was based on [Wat09, Remark 7.6], primarly through the use of the network's Taylor series and the Hilbert basis theorem. In Chapter 2, we showed that the Taylor series of a biasless tanh network of any depth is a sum of products of polynomials of the network weights and functions of the input. Using this form of the Taylor series, in Chapter 3 we derived a set of conditions based around the Hilbert basis theorem which guarantee a polynomial upper bound for K. Checking these conditions though was far from straightforward. To deal with these difficulties we introduced Gröbner bases and the generic division algorithm. Using these techniques, we checked the conditions for a two neuron network in Chapter 4. However, for larger networks, the generic division algorithm was too complicated. Instead, to prove the Main Theorem for a network with arbitrarily many neurons, we had to search for and use a pattern in the Taylor series polynomials to check all of the convergence conditions. Finally, we used the polynomials we found to visually examine the zero sets for the KL divergence of simple two layer networks in Chapter 5.

We began this project aiming to develop a general method for finding equivalent polynomials based on [Wat09, Remark 7.6]. This quickly proved out of reach, and instead we focused on understanding the specific example of two layer tanh networks which appear throughout the literature. While an important problem is to find equivalent polynomials for other types of networks, there may be several obstacles to generalising the methods from this thesis. For example, the generic division algorithm was already too difficult for models with more than two neurons, making it unlikely to work for deeper networks with even more parameters. The second method, which led to our result for two layer networks with arbitrarily many neurons, essentially involved being lucky enough to notice a pattern in the quotients. Hoping that the Taylor series polynomials are sufficiently "nice" for such a pattern to exist and be visible is not an ideal approach to rely on in general.

These limitations however, suggest interesting open problems and directions for future work. It would be worth trying some the methods in this thesis on other models, in order to confirm if they are in fact too difficult to generalise. By Lemma 5, we know that deeper tanh networks have polynomial Taylor series coefficients, so one option would be to try and prove a polynomial upper bound for a simple three layer network.

What caught our attention in [Wat09, Remark 7.6] though, is its purported generality. In its stated form, none of the detailed algebra and the conditions we had to check matter. Understanding whether such a general result is possible would be a fruitful problem to solve. For example, if one could prove that

$$\sum_{j=0}^{\infty} f_j(w)^2 < C \qquad \text{for all } w \in W$$

automatically implied the existence of quotients $a_j^k(w)$ and a constant $C' > 0$ such that

$$\sum_{j=0}^{\infty} a_j^k(w)^2 < C' \,,$$

then all the calculations in this thesis could be avoided. It would be much simpler to find polynomial upper bounds for any network whose Taylor series satisfy the conditions of the Replacement Strategy, potentially leading the way to calculations of the RLCTs for a much wider class of models than currently in the literature. At present though, we have no indication or intuition of whether such a statement is true.

Appendix A

Appendix

A.1 Other activation functions

In Chapter 2 we saw that biasless tanh networks have Taylor series coefficients which are polynomials of the weights and we proved that this holds for any network whose activation function satisfies the conditions in Lemma 4. In this Appendix, we examine which other activation functions have these properties, and show that Swish neural networks also satisfy the condition of the Replacement Strategy.

Sigmoid:

The logistic function

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

has derivative

$$\sigma'(x) = \sigma(x)(1-\sigma(x))\,,$$

and so σ satisfies property 2 of Lemma 4. However $\sigma(0) = \frac{1}{2}$ and so this activation doesn't satisfy property 1 of that lemma.

If we want, we can shift this downwards to invent a new activation that satisfies both properties 1 and 2. Define $\tau(x) = \sigma(x) - \frac{1}{2}$. Clearly $\tau(0) = 0$, and

$$\begin{aligned}\tau'(x) &= \sigma(x)(1-\sigma(x)) \\ &= \left(\frac{1}{2}+\tau(x)\right)\left(\frac{1}{2}-\tau(x)\right) \\ &= \frac{1}{4} - \tau(x)^2\,,\end{aligned}$$

so both properties hold. However $\tau(x) = \frac{1}{2}\tanh(\frac{1}{2}x)$, so this doesn't really add anything new.

Swish:

The Swish activation function is

$$s(x) = \frac{x}{1+e^{-\beta x}}\,,$$

for some fixed $\beta \in \mathbb{R}$. This activation satisfies property 1, and its derivative is

$$\begin{aligned}s'(x) &= \frac{(1+e^{-\beta x}) - x(-\beta e^{-\beta x})}{(1+e^{-\beta x})^2} \\ &= \frac{1}{1+e^{-\beta x}} + \frac{\beta x(1+e^{-\beta x}) - \beta x}{(1+e^{-\beta x})^2}\end{aligned}$$

$$= \sigma(\beta x) + \beta s(x) - \beta s(x)\sigma(\beta x)$$
$$= \sigma(\beta x)\left(1 - \beta s(x)\right) + \beta s(x)\,, \tag{A.1.1}$$

which is not a polynomial of $s(x)$ so property 2 does not hold. Despite this, the Taylor series coefficients for Swish networks are in fact polynomials of the weights. The derivative (A.1.1) is still a polynomial, but of both $s(x)$ and $\sigma(\beta x)$, which turns out to be sufficient.

We can replace σ by τ in (A.1.1), giving

$$\begin{aligned} s'(x) &= \left(\tau(\beta x) + \frac{1}{2}\right)\left(1 - \beta s(x)\right) + \beta s(x) \\ &= Q(s(x), \tau(\beta x))\,, \end{aligned}$$

where $Q(x_1, x_2) = \left(x_2 + \frac{1}{2}\right)(1 - \beta x_1) + \beta x_1$.

To prove that Swish neural networks do in fact have polynomial Taylor series coefficients, we introduce some definitions and preliminary lemmas.

Definition 27. Given a neural network F with Swish activations, we can write the output of F as

$$F(x, w) = s(z^{(n)})\,,$$

where $z^{(n)}$ is the n^{th} layer's weighted input. Define the *TauNet* of F to be the neural network T which has the same structure and weights as F but replaces the final layer activation function with $\tau(\beta(-))$, so that

$$\begin{aligned} T(x, w) &= \tau(\beta z^{(n)}) \\ &= \tau(\beta \sum_{j=1}^{k} w_{1j}^{(n)} a_j^{(n-1)})\,. \end{aligned}$$

The TauNet for F has the following property:

$$\begin{aligned} \frac{\partial T}{\partial x_i} &= \left(\frac{1}{4} - \tau(\beta z^{(n)})^2\right)\beta\left(\sum_{j=1}^{k} w_{1j}^{(n)} \frac{\partial a_j^{(n-1)}}{\partial x_i}\right) \\ &= \beta\left(\frac{1}{4} - T(x)^2\right)\left(\sum_{j=1}^{k} w_{1j}^{(n)} \frac{\partial a_j^{(n-1)}}{\partial x_i}\right) \\ &= G(T)\left(\sum_{j=1}^{k} w_{1j}^{(n)} \frac{\partial a_j^{(n-1)}}{\partial x_i}\right)\,, \end{aligned} \tag{A.1.2}$$

where $G(T) = \beta(\frac{1}{4} - T^2)$.

Definition 28. We say that a function is an *SP-function* for F if it is a polynomial of F, the TauNet for F, T, the network weights, $w_{ij}^{(l)}$, and the partial derivatives of the activations of the second last layer $\frac{\partial^{|\alpha|} a_j^{(n-1)}}{\partial x^\alpha}$. Equation (A.1.2) says that any first order partial derivative of the TauNet T is an SP function for F.

Lemma 18. Any partial derivative of an SP function for F with respect to an input x_i is itself an SP function for F.

Proof: It is sufficient to prove this is true for an SP "monomial"

$$w^\alpha F^a T^b \prod_{l=1}^{r} f_l\,, \tag{A.1.3}$$

where each f_l is a partial derivative of a second last layer neuron. Then

$$\frac{\partial}{\partial x_i}\left(w^\alpha F^a T^b \prod_{l=1}^{r} f_l\right) = w^\alpha \left(\prod_{l=1}^{r} f_l\right)\left[aF^{a-1}T^b Q(F,T) + bF^a T^{b-1} G(T)\right]\left(\sum_{j=1}^{k} w_{1j}^{(n)} \frac{\partial a_j^{(n-1)}}{\partial x_i}\right)$$
$$+ w^\alpha F^a T^b \sum_{l=1}^{r}\left(\frac{\partial f_l}{\partial x_i}\prod_{j\neq l} f_j\right),$$

which is an SP-function for F. □.

This lets us prove the polynomial coefficient property for Swish neural networks.

Lemma 19. Given a biasless neural network $F : \mathbb{R}^m \times W \to \mathbb{R}$ of depth n which uses Swish activation functions, the partial derivatives of F evaluated at $x = 0$ are polynomials of the weights.

Proof: We follow a similar strategy to Lemma 5, using induction on the network depth. First we prove the claim for 2 layer Swish networks of the form

$$F(x,w) = s\left(\sum_{j=1}^{m} w_{1,j}^{(2)} x_j\right). \tag{A.1.4}$$

This is an SP-function for F, so by Lemma 18, any partial derivative of F (up to any order) is again an SP-function for F and is a linear combination of terms of the form

$$w^\alpha F^a T^b x^\beta,$$

where α and β are multi-indices. The x^β factor appears since the activations of the second last layer (ie: layer 1) are just the inputs to the network. Then since $F|_{x=0} = 0$, $T|_{x=0} = 0$, and $x|_{x=0} = 0$, the above monomial restricted to $x = 0$ is a polynomial of the weights (which can be non-zero if a, b, and β are all zero). This proves the claim for two layer Swish networks.

As in the proof of Lemma 5, we use induction to prove the claim for networks of any depth. Suppose that the claim held for all Swish networks with n layers. If F is a swish network with $n+1$ layers, any partial derivative of F is an SP-function for F, meaning it is a linear combination of terms of the form

$$w^\alpha F^a T^b \prod_{l=1}^{r} f_l,$$

where each f_l is a partial derivative of an n^{th} layer's activation. By the induction hypothesis, each $f_l|_{x=0}$ is a polynomial of the network weights. It follows that any partial derivative of F evaluated at $x = 0$ is a polynomial of the weights. □.

A.2 Revisiting biases

In Lemma 5 we found that the Taylor series coeficients for biasless networks with certain activation functions are polynomials of the weights. We noted that including biases can make this result false. This causes a significant restriction, and understanding networks which have biases is an important problem which has not yet been studied in SLT.

The issue was that biases made a network's Taylor series coefficents be analytic functions of the weights, rather than just polynomials. In this Appendix we introduce a workaround which may help find a polynomial upper bound on the KL divergence of two layer tanh networks which have biases.

Suppose that we wish to find a polynomial upper bound for the KL divergence of the two layer tanh network with biases

$$f(x_1, a_1, b_1, c_1, ..., a_n, b_n, c_n) = \sum_{i=1}^{n} a_i \tanh\left(b_i x_1 + c_i\right),$$

where the input distribution is $q(x_1)$, and we assume the true function $f_T(x_1)$ is zero.

We can turn f into a biasless network by defining a *modified network* $\hat{f}$, which has an extra input dimension

$$\hat{f}(x_1, x_2, a_1, b_1, c_1, ..., a_n, b_n, c_n) = \sum_{i=1}^{n} a_i \tanh(b_i x_1 + c_i x_2) \ . \tag{A.2.1}$$

When x_2 is restricted to equal 1, $\hat{f}$ becomes the original network f. If we set the *modified input distribution* $\hat{q}(x_1, x_2)$ to equal $q(x_1)\delta(x_2 - 1)$, the KL divergence of the modified network is

$$\begin{aligned}
\hat{K}(a_1, b_1, c_1, ..., a_n, b_n, c_n) &= \int_{\mathbb{R}^2} \hat{f}(x_1, x_2, a_1, b_1, c_1, ..., a_n, b_n, c_n)^2 \delta(x_2) q(x_1) dx_2 dx_1 \\
&= \int_{\mathbb{R}} f(x_1, a_1, b_1, c_1, ..., a_n, b_n, c_n)^2 q(x_1) dx_1 \\
&= K(a_1, b_1, c_1, ..., a_n, b_n, c_n) \, ,
\end{aligned}$$

where $K(a_1, b_1, c_1, ..., a_n, b_n, c_n)$ is the KL divergence of the unmodified network f.

The modified network is a two layer biasless tanh network which has two inputs. By Lemma 5, the Taylor series coefficients for the modified network are polynomials of the weights. A worthwile future exercise, would be to check conditions (C1)-(C4) for these polynomials, hopefully resulting in a polynomial upper bound for a network with biases.

Remark. This is clearly related to finding a polynomial upper bound for the KL divergence of two layer tanh networks with multiple inputs.

In the paper [Wat01c], Watanabe uses this same construction to include networks with biases into his analysis. In the paper, he finds upper bounds on the RLCT of multidimensional networks based on the input and output dimensions, and number of neurons in the true function. His technique appears similar to the paper [Wat00c], where he replaces terms in the KL divergence with polynomials, but does not appear to resolve singularities.

Aoyagi and Watanabe have studied the RLCTs of two layer networks with both multiple inputs and outputs in [AW06], and their method relied on replacing the KL divergence with a polynomial. If we use their result to focus on networks with 1D outputs, their method would replace the KL divergence of the network with N inputs and H tanh units

$$f(x_1, ..., x_N, w) = \sum_{i=1}^{H} a_i \tanh\left(\sum_{j=1}^{N} b_{ij} x_j\right)$$

with the polynomial

$$\sum_{n=1}^{H} \sum_{j=1}^{N} \left(\sum_{i=1}^{H} a_i b_{ij}^{2(n-1)+1}\right)^2 .$$

Bibliography

[AW04] Miki Aoyagi and Sumio Watanabe. The generalization error of reduced rank regression in bayesian estimation. In *Proc. of ISITA*, pages 1068–1073, 2004.

[AW05] Miki Aoyagi and Sumio Watanabe. Resolution of singularities and the generalization error with bayesian estimation for layered neural network. *IEICE Trans.*, volume 88(10), pages 2112–2124, 2005.

[AW06] Miki Aoyagi and Sumio Watanabe. Generalization error of three layered learning model in bayesian estimation. In *Computational Intelligence*, pages 295–300, 2006.

[AW09] Miki Aoyagi and Sumio Watanabe. The zeta function for learning theory and resolution of singularities. In Heinrich G W Begehr and Francesco Nicolosi, editors, *More Progresses in Analysis*, pages 279–288. World Scientific, 2009.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[Car21] Liam Carroll. Phase transitions in neural networks. Master's thesis, The University of Melbourne, 2021. Available at http://therisingsea.org/notes/MSc-Carroll.pdf, accessed 1/4/2022.

[CLO15] David A. Cox, John B. Little, and Donal O'Shea. *Ideals, varieties, and algorithms : an introduction to computational algebraic geometry and commutative algebra.* Undergraduate texts in mathematics. Springer, 2015.

[DGPS21] Wolfram Decker, Gert-Martin Greuel, and Hans Schönemann, SINGULAR 4-2-1 — A computer algebra system for polynomial computations, http://www.singular.uni-kl.de, 2021.

[Fuk96] Kenji Fukumizu. A regularity condition of the information matrix of a multilayer perceptron network. Neural Networks, volume 9(5), pages 871–879, 1996.

[HTU93] Katsuyuki Hagiwara, Naohiro Toda, and Shiro Usui. On the problem of applying AIC to determine the structure of a layered feedforward neural network. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 3, pages 2263–2266, 1993.

[JEP+21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, volume 596(7873), pages 583–589, 2021.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[Lin11] Shaowei Lin. Algebraic methods for evaluating integrals in Bayesian statistics. PhD thesis, UC Berkely, 2011.

[LLPS93] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, volume 6(6), pages 861–867, 1993.

[LLS12] John Liu, Seymour Lipschutz, and Murray Spiegel. *Schaum's Outline of Mathematical Handbook of Formulas and Tables*, McGraw-Hill, USA, 4th edition, 2012.

[MWG+20] Daniel Murfet, Susan Wei, Mingming Gong, Hui Li, Jesse Gell-Redman, and Thomas Quella. Deep learning is singular, and that's good. *arXiv preprint arXiv:2010.11560*, 2020.

[Sus92] Héctor J. Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks*, volume 5(4), pages 589–593, 1992.

[TW20] Tadashi Takahashi and Tomohiro Washino. On the application of elimination ideal for statistical model. *COMPUSOFT, An International Journal of Advanced Computer Technology*, volume 9, pages 3685–3689, 2020.

[War21] Thomas Waring. Geometric perspectives on program synthesis and semantics. Master's thesis, The University of Melbourne, 2021. Available at https://thomaskwaring.github.io/thesis.pdf, accessed 1/4/2022.

[Wat99a] Sumio Watanabe. Algebraic analysis for neural network learning. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, volume 1, pages 431–436, 1999.

[Wat99b] Sumio Watanabe. Algebraic analysis for non-regular learning machines. In Sara Solla, Todd Leen, and Klaus Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 356–362. MIT Press, 1999.

[Wat99c] Sumio Watanabe. Algebraic analysis for singular statistical estimation. In Osamu Watanabe and Takashi Yokomori, editors, *Algorithmic Learning Theory. ALT 1999. Lecture Notes in Computer Science*, volume 1720, pages 39–50. Springer Berlin Heidelberg, 1999.

[Wat00a] Sumio Watanabe. Algebraic information geometry for learning machines with singularities. In Todd Leen, Thomas Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 311–317. MIT Press, 2000.

[Wat00b] Sumio Watanabe. Mathematical foundation for redundant statistical estimation. *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, volume 2000, pages 119–124, 2000.

[Wat00c] Sumio Watanabe. On the generalization error by a layered statistical model with bayesian estimation. *Electronics and Communications in Japan Part III-fundamental Electronic Science - ELECTRON COMMUN JPN III*, volume 83, pages 95–106, 2000.

[Wat01a] Sumio Watanabe. Algebraic Analysis for Nonidentifiable Learning Machines. *Neural Computation*, volume 13(4), pages 899–933, 2001.

[Wat01b] Sumio Watanabe. Algebraic geometrical methods for hierarchical learning machines. *Neural Networks*, volume 14(8), pages 1049–1060, 2001.

[Wat01c] Sumio Watanabe. Learning efficiency of redundant neural networks in bayesian estimation. *IEEE Transactions on Neural Networks*, volume 12(6), pages 1475–1486, 2001.

[Wat09] Sumio Watanabe. *Algebraic Geometry and Statistical Learning Theory*. Cambridge monographs on applied and computational mathematics. Cambridge University Press, 2009.

[Wat13] Sumio Watanabe. A widely applicable bayesian information criterion. *Journal of Machine Learning Research*, volume 14, pages 867–897, 2013.

[WT21] Tomohiro Washino and Tadashi Takahashi. Parametrization of statistical models in three-layer neural networks. In *2021 9th International Conference on Information and Education Technology (ICIET)*, pages 386–390, 2021.

[YG21] Dongbing Yu and Yu Gu. A machine learning method for the fine-grained classification of green tea with geographical indication using a mos-based electronic nose. Foods, volume 10(4), 2021.

[YW03a] Keisuke Yamazaki and Sumio Watanabe. Singularities in mixture models and upper bounds of stochastic complexity. *Neural Networks*, volume 16(7), pages 1029–1038, 2003.

[YW03b] Keisuke Yamazaki and Sumio Watanabe. Stochastic complexities of hidden markov models. In *2003 IEEE XIII Workshop on Neural Networks for Signal Processing (IEEE Cat. No.03TH8718)*, pages 179–188, 2003.