# Introduction

The purpose of this homework is to explain Lemma 3.1 and Lemma 3.2 on polynomial approximation by shallow tanh neural networks in a way that is accessible and clear. To achieve this, I have simplified some of the notation compared to the original paper, and added concrete examples to illustrate how the method works. The goal is not only to reproduce the technical statements, but also to highlight the main ideas step by step.

# Lemma 3.1: Odd Polynomials

**Lemma 1** (Approximation of Odd Polynomials)**.** *Let $k \in \mathbb{N}_0$ and $s \in 2\mathbb{N} - 1$. For every $\epsilon > 0$, there exists a one-hidden-layer neural network $\Psi_{s,\epsilon} : [-M, M] \to \mathbb{R}^{\frac{s+1}{2}}$ with activation* tanh *and width about $\frac{s+1}{2}$ such that*

$$\max_{\substack{p \le s \\ p \text{ odd}}} \left\| f_p - (\Psi_{s,\epsilon})_{\frac{p+1}{2}} \right\|_{W^{k,\infty}} \le \epsilon.$$

## Key Ideas

**1. Taylor expansion of** tanh**.** Near zero,

$$\tanh(z) = z - \tfrac{z^3}{3} + O(z^5).$$

Hence, for small $h$,

$$\tanh(hx) = hx - \tfrac{(hx)^3}{3} + O(h^5),$$

and dividing by $h$ yields

$$L_h(x) := \frac{1}{h} \tanh(hx) = x - \tfrac{h^2}{3} x^3 + O(h^4 x^5).$$

$$L_h(x) - x = -\tfrac{h^2}{3} x^3 + O(h^4 x^5).$$

Taking absolute values and restricting to $|x| \le 1$ gives

$$|L_h(x) - x| \le \tfrac{h^2}{3} + O(h^4).$$

Thus $L_h(x)$ approximates $x$ with error of order $O(h^2)$ on $[-1, 1]$.

**2. Finite difference operator.** To extract higher-order information, we introduce the finite difference operator

$$\delta_h^p[f](x) := \sum_{i=0}^{p} (-1)^i \binom{p}{i} f\left(x + \left(\tfrac{p}{2} - i\right)h\right).$$

This operator evaluates $f$ at equally spaced points around $x$ with alternating binomial weights. Its design ensures that lower-order terms in the Taylor expansion cancel, while the $p$-th order term survives. In general,

$$\delta_h^p[f](x) = h^p f^{(p)}(x) + O(h^{p+2}).$$

**Example ($p = 2$).** For $p = 2$ we have

$$\delta_h^2[f](x) = f(x+h) - 2f(x) + f(x-h).$$

Expanding in Taylor series:

$$f(x+h) = f(x) + hf'(x) + \tfrac{h^2}{2}f''(x) + \tfrac{h^3}{6}f^{(3)}(x) + O(h^4),$$

$$f(x-h) = f(x) - hf'(x) + \tfrac{h^2}{2}f''(x) - \tfrac{h^3}{6}f^{(3)}(x) + O(h^4).$$

Adding these and subtracting $2f(x)$ gives

$$\delta_h^2[f](x) = h^2 f''(x) + O(h^4).$$

**Interpretation.** The constant term $f(x)$ cancels, the first derivative terms $f'(x)$ also cancel, and only the second derivative $f''(x)$ remains (up to higher-order error). Thus the finite difference operator acts like a "filter" that isolates the $p$-th derivative. In Lemma 3.1, this property is used to construct approximations of odd powers $x^p$ by combining tanh activations at different scales and locations.

**3. Katsuura identity.** A crucial ingredient in the analysis is the Katsuura identity:

$$\sum_{i=0}^{p} (-1)^i \binom{p}{i} \left(\tfrac{p}{2} - i\right)^{\ell} = \begin{cases} p!, & \ell = p, \\ 0, & \ell \neq p. \end{cases}$$

**Meaning.** When $f(x + (\frac{p}{2} - i)h)$ is expanded in a Taylor series, terms of the form $(\frac{p}{2} - i)^{\ell}$ naturally appear:

$$f(x+sh) = f(x) + shf'(x) + \tfrac{(sh)^2}{2}f''(x) + \cdots + \tfrac{(sh)^{\ell}}{\ell!}f^{(\ell)}(x) + \cdots.$$

Plugging this into the definition of the finite difference operator leads to sums of the form

$$\sum_{i=0}^{p} (-1)^i \binom{p}{i} \left(\tfrac{p}{2} - i\right)^{\ell}.$$

The Katsuura identity guarantees that for $\ell < p$ these sums vanish, for $\ell = p$ the sum equals $p!$, and for $\ell > p$ the terms contribute only to the higher-order error.

**Example ($p = 2$).**

$$\sum_{i=0}^{2} (-1)^i \binom{2}{i} (1-i)^{\ell}.$$

For $\ell = 0$, the sum is $1 - 2(1) + 1 = 0$. For $\ell = 1$, the sum is $1 - 0 - 1 = 0$. For $\ell = 2$, the sum is $1 + 1 = 2 = 2!$. Again, the identity holds.

**Interpretation.** The Katsuura identity is the "engine" behind the finite difference operator: it ensures that $\delta_h^p$ cancels all lower-order contributions and retains only the $p$-th derivative. This property is exactly what allows us to construct pure odd powers $x^p$ from combinations of tanh activations in Lemma 3.1.

## 4. Error estimate.

**Error bound.** For an odd polynomial $f_p$ of degree $p$, let $\widehat{f}_{p,h}$ denote its approximation constructed using tanh activations and the finite difference operator. Then one can estimate

$$\|f_p - \widehat{f}_{p,h}\|_{W^{k,\infty}} \; \le \; C_{p,k,M}\, h^2,$$

where $C_{p,k,M}$ is a constant depending only on the degree $p$, the smoothness order $k$, and uniform bounds $M$ on higher derivatives of tanh. The quadratic order $h^2$ arises because the symmetric structure of the operator cancels additional low-order error terms.

**Choosing $h$.** Given a tolerance $\epsilon > 0$, it suffices to pick

$$h \; \le \; \sqrt{\tfrac{\epsilon}{C_{p,k,M}}}$$

so that the approximation error is controlled by $\epsilon$.

**Interpretation.** The error decreases quadratically in $h$, so by shrinking the scaling parameter $h$ we can make the approximation arbitrarily accurate.

**5. Weight bound.** In the previous section we observed that to control the approximation error, one must choose a small scaling parameter $h$. However, the finite difference operator produces

$$\delta_h^p[f](x) = h^p f^{(p)}(x) + O(h^{p+2}),$$

so in order to recover the correct scale of $f^{(p)}(x)$, one must divide by $h^p$. This division manifests in the neural network as large output weights, especially when $h$ is very small.

**Contribution of binomial coefficients.** The coefficients of the finite difference operator also contain binomial terms:

$$\sum_{i=0}^{p} (-1)^i \binom{p}{i} f\Big(x + \big(\tfrac{p}{2} - i\big)h\Big).$$

Thus the network weights are influenced not only by $1/h^p$ but also by the size of $\binom{p}{i}$, which can be large when $p$ is large.

**Stirling's approximation.** The binomial coefficient is maximized at $i \approx p/2$ and satisfies

$$\binom{p}{i} \le \binom{p}{\lfloor p/2 \rfloor} \sim \frac{2^p}{\sqrt{\pi p/2}}.$$

Hence one obtains the general upper bound

$$\binom{p}{i} \; \lesssim \; \Big(\tfrac{2p}{\sqrt{p}}\Big)^p.$$

**Final weight estimate.** Combining the $1/h^p$ factor with the binomial growth, the overall weight magnitude can be bounded by

$$\|\text{weights}\| = O\Big(\epsilon^{-s/2}\,(2(s+2)\sqrt{2M})^{\,s(s+3)}\Big).$$

Here the factor $\epsilon^{-s/2}$ reflects the trade-off between accuracy and parameter size (smaller $\epsilon$ requires larger weights). The exponential term

$$(2(s+2)\sqrt{2M})^{s(s+3)}$$

arises from two sources: the accumulation of binomial coefficients, which grow rapidly with $s$, and the uniform bounds $M$ on higher derivatives of tanh, which enter through Taylor expansions and difference formulas. This shows that although the weights can grow quickly with $s$ and $\epsilon^{-1}$, they remain explicitly bounded and therefore controllable.

**Interpretation.** This result shows that while smaller $h$ improves the approximation error, it also forces the network weights to grow. Nevertheless, the growth remains explicitly bounded: the weights depend polynomially and exponentially on $s$ (the maximum degree) and on $\epsilon$ (the desired accuracy), but do not blow up uncontrollably. Thus Lemma 3.1 provides not only an approximation guarantee but also a concrete complexity bound on the parameters.

### Example: Approximating $x$

Define

$$L_h(x) = \frac{1}{h}\tanh(hx).$$

Then

$$L_h(x) - x = -\tfrac{h^2}{3}x^3 + O(h^4x^5),$$

so that

$$\sup_{|x|\le 1} |L_h(x) - x| \le \tfrac{h^2}{3} + O(h^4).$$

By choosing $h \sim \sqrt{\epsilon}$, we achieve accuracy $\epsilon$.

## Lemma 3.2: Even Polynomials

**Lemma 2** (Approximation of Even Polynomials)**.** *Let $k \in \mathbb{N}_0$, $s \in 2\mathbb{N}-1$, and $M > 0$. For every $\epsilon > 0$, there exists a one-hidden-layer neural network $\psi_{s,\epsilon} : [-M, M] \to \mathbb{R}^s$ with activation* tanh *and width about $\frac{3(s+1)}{2}$ such that*

$$\max_{p\le s} \left\| f_p - (\psi_{s,\epsilon})_p \right\|_{W^{k,\infty}} \le \epsilon.$$

### Key Ideas

**1. Identity for even powers.** The starting point of Lemma 3.2 is the identity

$$x^{2m} = \frac{1}{2m+1}\Big[(x+1)^{2m+1} - (x-1)^{2m+1}\Big].$$

**Meaning.** This formula shows that an even power $x^{2m}$ can be expressed as a linear combination of odd powers. Since Lemma 3.1 already guarantees that odd powers can be approximated by shallow tanh-networks, this identity provides a direct way to approximate even powers as well.

**Example ($m = 1$).** For $m = 1$, the identity becomes

$$x^2 = \frac{(x+1)^3 - (x-1)^3}{6}.$$

Indeed,

$$(x+1)^3 - (x-1)^3 = (x^3 + 3x^2 + 3x + 1) - (x^3 - 3x^2 + 3x - 1) = 6x^2.$$

Thus the formula is correct, and it shows explicitly how $x^2$ can be written in terms of cubic polynomials.

**2. Recursive definition.** Equation (27) provides a recursive relation for constructing higher-order even powers. In general, an even power $x^{2m}$ can be expressed as a linear combination of

$$x^{2m-2} \quad \text{(a lower-order even term)} \qquad \text{and} \qquad \text{certain odd powers.}$$

**Meaning.** This recursion allows us to build $x^{2m}$ step by step: - start with $x^2$ (which can be obtained from identity (25)), - then use it together with odd powers to construct $x^4$, - then use $x^4$ to help construct $x^6$, and so on.

**3. Error estimate.** In Lemma 3.1 we already showed that odd powers can be approximated with error of order $O(h^2)$. Lemma 3.2 extends this to even powers. The construction of $x^{2m}$ uses two ingredients: a lower-order even power $x^{2m-2}$ and odd powers (approximable by Lemma 3.1). This raises the natural question: does the recursive construction increase the error?

**Intermediate steps (28)–(33).** The proof proceeds in several stages:

- (28): Define an explicit upper bound $E_p^*$ for the error $E_p$,

  $$E_p \le E_p^* := \frac{2^{p/2}(1+\alpha)^{(p^2+p)/2}}{\alpha^{p/2}} \cdot \epsilon,$$

  where $\alpha > 0$ is a free parameter that can be chosen to optimize the bound.

- (29): Show that with the choice of $h$ from Lemma 3.1, all *odd* degrees $p$ already satisfy $E_p \le \epsilon$. Thus the odd case is settled.

- (30): Establish the base case for even powers, namely $p = 2$, by proving

  $$E_2 \le \tfrac{1}{6\alpha} \cdot 2\epsilon \;\le\; E_2^*.$$

- (31)–(33): Use the recursive relation (27) together with identity (25) to express $x^{2m}$ in terms of $x^{2m-2}$ and odd powers. Applying the error bounds from Lemma 3.1 for odd powers, and using the sub-additivity of Sobolev norms, the error at step $2m$ is controlled in terms of the error at step $2m-2$. This inductive argument ensures that the error does not grow beyond $E_p^*$.

Together these arguments ensure that the recursive construction for even powers preserves the same order of accuracy as in the odd case.

**Final estimate (34).** The main conclusion is

$$\| x^{2m} - \widehat{x^{2m}} \|_{W^{k,\infty}} \ \leq \ \epsilon,$$

where $\widehat{x^{2m}}$ denotes the network approximation obtained from the recursive construction. This shows that the approximation error for even powers can be kept within the same tolerance $\epsilon$ as in the odd case.

**Interpretation.** The key reason the error remains controlled is that the recursion is linear:

$$x^{2m} = \alpha \cdot x^{2m-2} + \beta \cdot (\text{odd powers}),$$

and Sobolev norms behave well under linear combinations. As a result, the overall error remains of order $O(h^2)$, and the approximation for even polynomials is just as accurate as for odd polynomials.

**4. Weight bound.** Finally we analyze the size of the network weights required for approximating even powers. As in Lemma 3.1, the finite difference construction produces terms of the form $h^p f^{(p)}(x)$, which must be normalized by dividing by $h^p$. This introduces large factors when $h$ is small, and in addition the binomial coefficients from the operator contribute further growth. For even powers, the recursive construction from (27) leads to a slightly different but comparable bound.

**Final estimate (36).** The size of the weights satisfies

$$\|\text{weights}\| \ = \ O\left(\epsilon^{-s/2} \left(\sqrt{M}(s+2)\right)^{\frac{3}{2}s(s+3)}\right).$$

**Comparison with Lemma 3.1.** For odd powers the weight bound was

$$O\left(\epsilon^{-s/2} \left(2(s+2)\sqrt{2M}\right)^{s(s+3)}\right).$$

The even case (36) has the same structure: a precision factor $\epsilon^{-s/2}$ multiplied by polynomial and derivative factors depending on $s$ and $M$. The only difference is in the exponent, due to the recursive nature of the even-power construction. Thus both odd and even powers admit explicit, finite, and controllable weight bounds.

## Conclusion

In this homework we have discussed Lemma 3.1 and Lemma 3.2, which together establish that both odd and even polynomials can be approximated by shallow neural networks with tanh activation. Lemma 3.1 showed that odd powers of $x$ admit such an approximation with controlled error, while Lemma 3.2 extended the argument to even powers by a recursive construction.

Taken together, these results imply that *any polynomial up to degree $s$* can be approximated arbitrarily well by a one-hidden-layer tanh network of explicitly bounded width and weights. Since polynomials form a dense class in the space of continuous functions, this provides a constructive path toward the universal approximation property of shallow neural networks.