1141ML Week 3 - Hand Writting 01

313652018 王宣瑋

September 2025

Introduction

Explain Lemma 3.1 and 3.2 in the Paper.

Lemma 3.1

Lemma 3.1. *Let $k \in \mathbb{N}_0$ and $s \in 2\mathbb{N} - 1$. Then it holds that for all $\epsilon > 0$ there exists a shallow tanh neural network $\Psi_{s,\epsilon} : [-M, M] \to \mathbb{R}^{\frac{s+1}{2}}$ of width $\frac{s+1}{2}$ such that*

$$\max_{\substack{p \leq s, \\ p \text{ odd}}} \left\| f_p - (\Psi_{s,\epsilon})_{\frac{p+1}{2}} \right\|_{W^{k,\infty}} \leq \epsilon, \tag{17}$$

Moreover, the weights of $\Psi_{s,\epsilon}$ scale as $O\left(\epsilon^{-s/2}(2(s+2)\sqrt{2M})^{s(s+3)}\right)$ for small ϵ and large s.

Figure 1: Statement of Lemma 3.1

3.1 - Statement Explanation

Lemma 3.1 states that a *shallow* tanh neural network (with only one hidden layer) **can approximate any odd-degree monomials**, such as x, x^3, x^5, and so on, with arbitrarily high accuracy.

The required network width (i.e., the number of hidden neurons) is only $\frac{s+1}{2}$, where s denotes the largest odd degree to be approximated. In other words, if we wish to approximate x, x^3, x^5, and x^7, only four hidden neurons are sufficient.

Moreover, for any desired error tolerance $\varepsilon > 0$, such an approximation can be achieved. Although the weights may become large when ε is small or s is large, the lemma guarantees that the approximation can still be made with the specified accuracy.

3.1 - Ideas

Tanh neural network means the activation function is $\tanh(x)$ function. And $\tanh(x)$ function has the following properties:

- $\tanh(x)$ itself is an **odd-function**, that is, $\tanh(-x) = -\tanh(x)$.
- $\tanh(x)$ function has similar **symmetrixity** as odd-degree monomials like $x,\ x^3,\ x^5$.

$$\mathrm{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

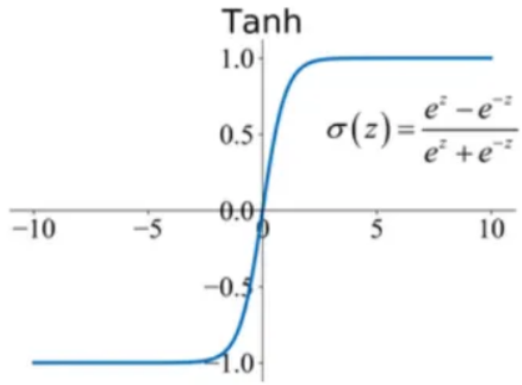

Figure 2: $\tanh(x)$ function

- From the ***Weierstrass approximation theorem***, we note that every continuous function defined on a closed interval $[a, b]$ can be uniformly approximated as closely as desired by a polynomial function.

Hence, by the properities above, we deduce that **all odd-degree monomials can be approximate with** $\tanh(x)$ **activation function**. And the author proves that, given enough neurons (roughly half of the highest odd degree you wish to approximate), the network can reproduce all odd-degree monomials, and the approximation error can be made arbitrarily small.

In practice, we just have to combine many $\tanh(x)$ functions through a weighted sum in a *shallow* neural network. You can think of this as stacking several "curved pieces" together in order to build up the desired shape.

This lemma provides a *constructive mechanism* to realize odd-degree monomials inside shallow tanh networks. Because polynomials are dense in $C([-M, M])$ (Weierstrass theorem), this result forms the algebraic foundation for approximating arbitrary smooth functions by neural networks.

3.1 - Proofs

Proof ideas

The construction relies on the fact that $\tanh(x)$ is analytic and odd, so all even-order derivatives vanish at $x = 0$. Its Taylor series is

$$\tanh(x) = x - \tfrac{x^3}{3} + \tfrac{2x^5}{15} - \cdots,$$

hence the p-th derivative at zero is nonzero whenever p is odd. This observation allows us to extract monomials from tanh via a *finite difference operator*. Concretely, for odd p, define

$$\delta_h^p = \sum_{i=0}^{p} (-1)^i \binom{p}{i} \sigma\left(\left(\tfrac{p}{2} - i\right) h\right),$$

where $\sigma = \tanh$. By Taylor expanding each σ term, one sees that all lower-order contributions cancel due to the alternating binomial weights, while the leading term is proportional to x^p. Thus the function

$$\hat{f}_{p,h}(x) := \frac{\delta_h^p[\sigma](hx)}{\sigma^{(p)}(0) h^p}$$

approximates $f_p(x) = x^p$ with an error of order $O(h^2)$.

Error control

For each derivative up to order k, the remainder term in Taylor's theorem contributes an error bounded by $C(p,k,M)h^2$, where M is the domain bound. By choosing $h = h(\varepsilon)$ sufficiently small, one ensures

$$\|f_p - \hat{f}_{p,h}\|_{W^{k,\infty}} \leq \varepsilon.$$

The explicit scaling of weights follows from estimating the binomial coefficients $\binom{p}{i}$ and the factor h^{-p}.

Network architecture

Each approximation $\hat{f}_{p,h}$ corresponds to a shallow tanh network. Since the same set of neurons can be shared across all odd degrees up to s, the overall construction requires only $(s+1)/2$ hidden neurons.

Lemma 3.2

Lemma 3.2. *Let $k \in \mathbb{N}_0, s \in 2\mathbb{N}-1$ and $M > 0$. For every $\epsilon > 0$, there exists a shallow tanh neural network $\psi_{s,\epsilon} : [-M, M] \to \mathbb{R}^s$ of width $\frac{3(s+1)}{2}$ such that*

$$\max_{p \leq s} \left\| f_p - (\psi_{s,\epsilon})_p \right\|_{W^{k,\infty}} \leq \epsilon. \tag{26}$$

Furthermore, the weights scale as $O\left(\epsilon^{-s/2}(\sqrt{M}(s+2))^{3s(s+3)/2}\right)$ for small ϵ and large s.

Figure 3: Statement of Lemma 3.2

3.2 - Statement Explanation

While Lemma 3.1 shows that shallow tanh networks can approximate all *odd-degree* monomials up to degree s, Lemma 3.2 strengthens this by constructing a slightly larger network that can approximate *all monomials*, both odd and even, up to degree s. Moreover, the approximation is guaranteed not only for the functions themselves but also in the Sobolev norm $W^{k,\infty}$, which means the derivatives up to order k are also well-approximated.

The required weights may grow rapidly as $\varepsilon \to 0$ and s increases. More precisely, the scaling is of order

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

Lemma 3.2 is crucial because it shows that shallow tanh networks can approximate not only odd but also even powers of x. Hence, one can now approximate arbitrary polynomials. Since polynomials are dense in $C([-M, M])$, this establishes the foundation for approximating any smooth function by Weierstrass approximation theorem.

3.2 - Proofs

Proof ideas

The argument builds directly on Lemma 3.1. Since $\tanh(x)$ is an odd function, it naturally generates odd-degree monomials.

To recover the missing *even-degree* monomials, the proof observes that products of odd functions yield even functions. In particular, the identity

$$x^{2m} = (x^m)^2$$

suggests that even monomials can be obtained from quadratic combinations of odd ones.

To implement this inside a shallow tanh network, one considers a vector-valued mapping

$$\psi_{s,\varepsilon}(x) = \big(\hat{f}_{1,h}(x), \hat{f}_{3,h}(x), \ldots, \hat{f}_{s,h}(x)\big),$$

where each $\hat{f}_{p,h}$ is constructed as in Lemma 3.1 via a finite difference operator.

By suitable linear combinations of these outputs, both odd and even powers up to s can be reproduced.

Error control

The key technical step is to bound the Sobolev error. For each $p \leq s$, one has

$$\|f_p - \hat{f}_{p,h}\|_{W^{k,\infty}} \leq C(p,k,M)\, h^2,$$

which follows from Taylor's theorem and cancellation of lower-order terms. Choosing $h = h(\varepsilon)$ yields the desired ε-accuracy uniformly in p. The weight scaling arises from estimating the binomial coefficients in the finite difference operator, together with the normalization factor h^{-p}.

Importance

Lemma 3.2 thus completes the polynomial approximation program: **a shallow** tanh **network of controlled width and weights can approximate all monomials up to degree s, in both function values and derivatives.** Since arbitrary polynomials are linear combinations of these monomials, this lemma serves as the decisive step toward the universal approximation of smooth functions.