WrittenAssignment3_111652026劉馥瑞

(1)

1. Background

Neural networks are mathematical models that can approximate complicated functions. One way to judge how good a neural network is comes down to the following question: Can we use a neural network to approximate simple building blocks, like powers of x? For example, can we build a network that outputs something very close to x^3 or x^4 on an interval like [-M, M]? If we can approximate these simple functions well, then in principle we can combine them to approximate much more complicated functions (since polynomials can approximate a huge class of functions). The activation function in this setting is the tanh function (tanh(x)), which is a smooth S-shaped curve. Because tanh(x) is smooth and has useful symmetry properties, it can be combined in clever ways to approximate powers of x. The two lemmas we are studying (Lemma 3.1 and Lemma 3.2) answer this question for odd powers (like x, x^3, x^5, …) and even powers (like x^2, x^4, x^6, …).

2. Lemma 3.1 (Approximating Odd Powers)

Statement (in plain words): Suppose you want to approximate odd powers x^p (with p odd, like 1, 3, 5, …). Then, for any small error tolerance ε > 0, there exists a shallow neural network (a network with just one hidden layer) using the tanh activation function that gets within ε of the true function. The number of neurons needed in the hidden layer is about half of the degree you want: (s+1)/2 if you want to go up to degree s. Intuition: Odd powers are easier to approximate because tanh itself is an odd function (tanh(-x) = -tanh(x)). By shifting and scaling tanh, and then combining them in the right way, we can mimic the shape of x, x^3, x^5, etc. Think of this like using Lego blocks: each shifted tanh curve is a block, and with enough blocks you can build a curve that looks like x^3. Example (visual idea): To approximate x^3, imagine plotting tanh curves centered at different points, flipping some of them upside down, and then adding them together. The resulting curve bends in the same way as x^3.

3. Lemma 3.2 (Approximating Even Powers)

Statement (in plain words): Suppose you want to approximate even powers x^p (like x^2, x^4, x^6, …). These can also be approximated by shallow tanh networks, but the construction is a bit more complicated. The required network is larger than for the odd case: about 3(s+1)/2 neurons in the hidden layer. Again, the error can be made as small as you like. Intuition: Even powers are trickier because tanh is an odd function, and an odd function alone cannot directly produce even behavior. The trick is to build even powers out of odd powers. For example, a clever identity shows that you can write y^2 in terms of (y+α)^3, (y-α)^3, and lower powers. This means: if we already know how to approximate cubes (odd powers), we can recycle that construction to approximate squares (even powers). By repeating this process recursively, you can approximate all even powers using only combinations of odd-power approximations. Example (visual idea): To approximate x^2, notice that (x+1)^3 - (x-1)^3 ≈ 6x^2. So a difference of two cubes already produces a

parabola shape. This is the same idea generalized: use differences of shifted odd powers to build even powers.

4. Why These Lemmas Matter

a. Building blocks of polynomials: Together, Lemma 3.1 and Lemma 3.2 show that both odd and even powers can be approximated by shallow tanh networks. Since polynomials are made of sums of these powers, this means polynomials can be well-approximated too.

b. Towards universal approximation: Polynomials themselves can approximate a wide range of continuous functions (by results like the Weierstrass approximation theorem). So, if neural networks can approximate monomials, they can ultimately approximate any continuous function, at least in principle.

c. Efficiency: The lemmas don't just say approximation is possible—they also quantify how the network size (width) and weights (scaling) grow as you want higher accuracy or higher degree. This helps us understand the trade-off between approximation power and network complexity.

5. Summary (Student-Friendly Takeaway)

- Lemma 3.1: Odd powers (x, x^3, x^5, …) can be approximated by shallow tanh networks of width (s+1)/2. This works because tanh itself is odd.

- Lemma 3.2: Even powers (x^2, x^4, …) can be approximated too, by recycling odd-power approximations through algebraic identities. The networks needed are a bit bigger (width 3(s+1)/2). - Big picture: Together, these results show that shallow tanh networks can approximate all monomials up to a given degree, and therefore can serve as good approximators for general functions.

(2)

Why do we often take the log-likelihood instead of working directly with the likelihood function?