Lecture 18: Neural Networks - Representation Basics

Instructor: Aditya Bhaskara *Scribe: Pavan Kalyan Tummala*

CS 5966/6966: Theory of Machine Learning

March 17th, 2022

Abstract

In today's lecture we went through the basics of Neural Networks and Deep Learning. We also covered the depth-width trade-off and finally showing the power of depth.

1 Introduction

Neural Networks are a complicated hypothesis class that takes an input x and produces an output y. The main goal is to solve the ERM problem for this hypothesis class.

Given data $(x_1, y_1), (x_2, y_2), ...$ for a distribution D. Find the hypothesis class h that minimizes the risk. Where x_i is a feature vector $\in \mathbb{R}^n$ and y_i is the label $\in \mathbb{R}$.

For binary classification:

(1) $risk(h) = \mathbf{E}_{x\sim D}\mathbf{1}[h(x) \neq y]$

For real values:

(2) $risk(h) = \mathbf{E}_{(x,y)\sim D} l(h(x), y)$

Here l is the loss function.

This ERM is called training using the given data. Finding best hypothesis $(f(x_i) = y_i)$ for all i.

2 Theory of Deep Learning

Expressibility:

Expressibility is more of a structural questions.

Few examples are
How rich is the hypothesis class?

What kind of functions can be obtained using a DNN?

Training complexity and dynamics:

After solving an ERM problem using neural networks we need to check how good is the network on unseen test data.

Generalization:

What kind of generalization bounds can we prove ?
Example is VC dimension, number of parameters.

Answers for the questions are easy but are extremely incomplete in a realistic settings.

3 Expressibility Basics

Barron's Theorem

3.1 definition. Any continuous function f that satisfies an appropriate "niceness" condition that is parameterized by C can be approximated to an error ϵ by a 2-layer NN with $C^2/2$ internal nodes.

Nice functions can be approximated by small NN's.

Universal Approximation

3.2 definition. Any continuous function over a compact domain can be approximated by a 2-layer NN with any non-linearity.

Let's say we have the input dimension n the width of the neural network needed is 2^n. It is exponential. Hence, In practice 2 layer NN's are not used.

4 Depth vs Width

Depth allows the neural networks to learn meaningful features whereas width allows the neural networks to memorize the features.

In Practice deeper networks are used more frequently compared to wider networks. Width of a network comes into play in case we have a bunch of classes that are unrelated to each other. Different neurons will be connected to different parts to memorize the input.
Universality results degrade rapidly with dimensions. For example, the curse of dimensionality (width needs to be in exponential dimensions).

Example of Depth Width tradeoff

Let,

F_1 : {f : f is the output of depth 3 NN with width $\leq$ 100}

F_2 : {f : f is the output of depth 2 NN with width $\leq 10^8$}

The input size of the neural networks is 1.

The question is can the function in F_1 be approximated to an error $\leq \epsilon$ using a function in F_2?

There is a field in complexity theory that is circuit lower bound.

Circuit Lower Bounds:

Look at specific f of interest and ask if there exists a circuit of size $\leq$ S that computes f.

[Minsky and Pappert '69] showed that the parity function requires exponential width to do a Neural Network. A parity function outputs 0 if there are even number of 1's and outputs 1 if there are are odd number of 1's in the given binary string. This was one of the reasons to loose interest in Neural Networks.

5 Power of Depth

Template of Theorem: There exists a network of depth D and size S that computes some function f that cannot be approximated by the output of any network with depth d and sizes S'. Typically d << D and S' >> S.

Minsky showed hat if we have networks with ReLU activation then for any integer k there exists some network with D = k^3 a constant width. If we want to approximate it with a network with depth k then the width of the network would be exponential in k (i.e) exp(k).

Proof Outline:

The proof goes by considering one-dimensional inputs and ReLU activations. Proof is also divided into three parts.

Insights: depth D lets us achieve exp(D) oscillations in f. For width w and depth d $\#oscillations \leq w^{o(d)}$. In order to achieve so many oscillations with d requires a huge width.

If we want D = k^3 and $d = k$.

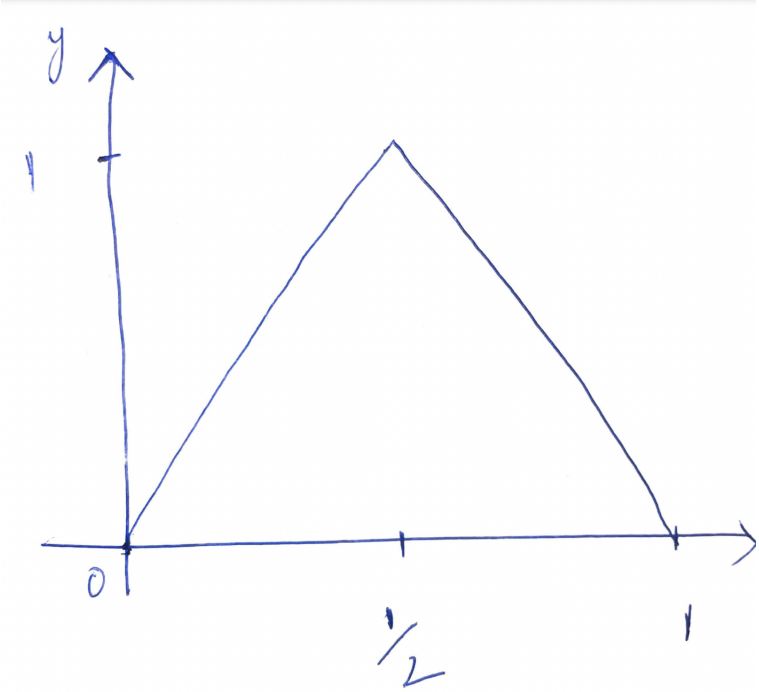
To match the number of oscillations:
$2^{k^3} = w^k$

$w > 2^{k^2}$

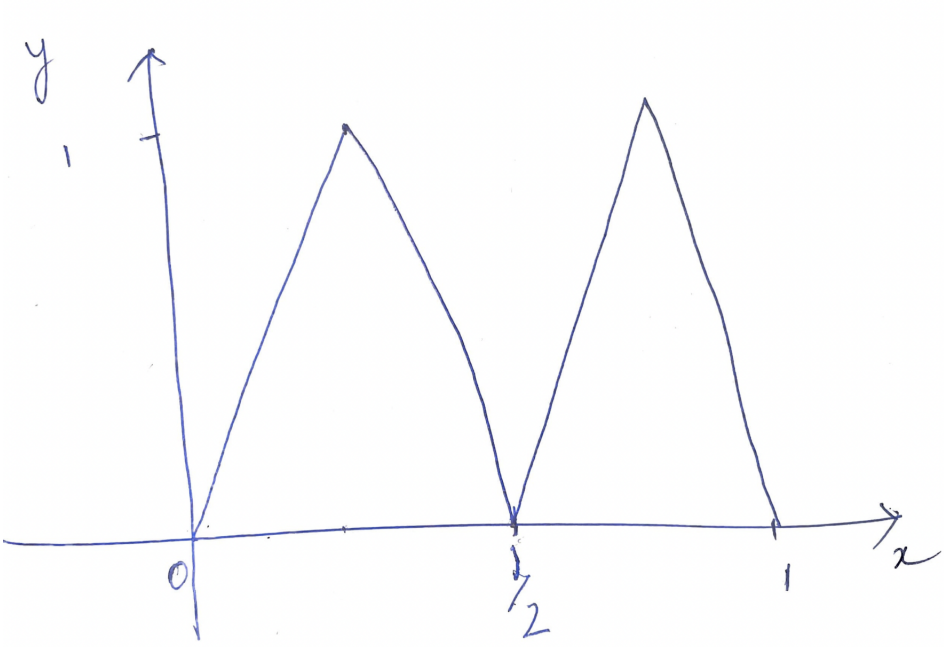
Part-1:

Let's take a single variable input x in domain [0,1]
let σ be a ReLU operator $\sigma(x) = max(0, x)$

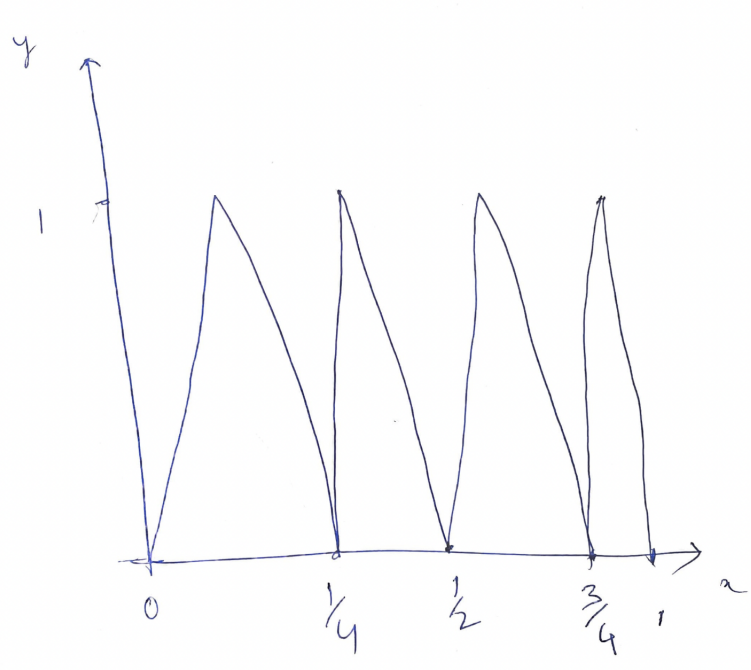
For the function below f(x) = $\sigma(2x) - \sigma(4(x - 1/2)) + \sigma(2(x - 1))$.



f(f(x)) looks like below and has 2 peaks.



f(f(f(x))) has 4 peaks and so on.



$f^{o(k+1)}(x)$ has 2^k peaks that means using a depth 2k NN with width ≤ 3 we can implement this function.

Part-2:

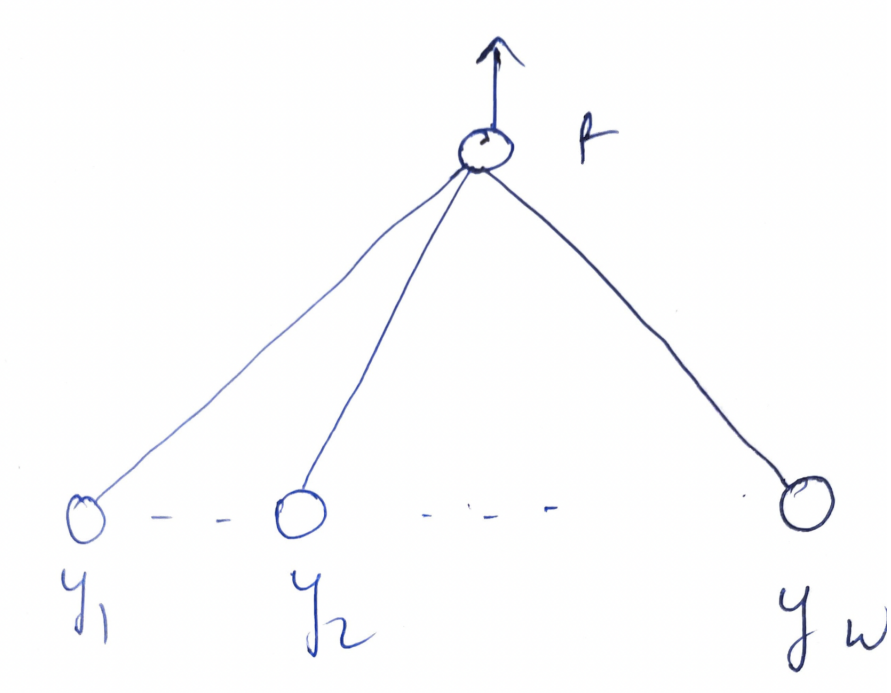
If we use depth $<<$ k getting this function requires a huge width.

Observation:

Let y_i be a neuron that is a piecewise linear function of variable x with $\leq n_i$ pieces.

Idea:

If we have a layer with neurons $y_1, y_2, y_3, ...y_w$ and the output layer is f such that f = $y_1 + y_2 + y_3 + ... + y_w$



We can say that f is also piecewise linear and the number of pieces in f is pretty small $\leq (n_1 + n_2 + ... + n_w)^2$.

Part-3:

Let's say f_1 and f_2 are piecewise linear function with m_1 and m_2 pieces and $m_1 \leq m_2/2$.

Then, $||f_1 - f_2|| \geq 1/8$. This shows that f_1 cannot be approximated to f_2.