Introduction

- Perceptrons are incapable of representing a rich set of functions (Minsky and Papert 1969)
- Feed forward networks perform well on a range of tasks
- Does this reflect an inherent ability of FF-NNs to represent arbitrary functions, or are they limited to a small set of functions that we just happen to have tested them on, albeit larger than the set of functions perceptrons can express.
- NN fans use Kolmogorov's superposition theorem to justify using neural networks
    - This states that every multivariate continuous function can be expressed as a finite composition of continuous functions of a single variable and the operation of addition:

$$f(\mathbf{x}) = \sum_{q=0}^{2n} \Phi \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right).$$

    - BTW this looks a bit like a Deep Set
    - But this requires a different unknown transformation for each f, we use the same activation functions for multiple f's! The theorem states a particular upper bound on hidden units needed for representation, in NNs we increase the number of hidden units until we get good performance.
    - NN's also put constraints on the smoothness (continuous derivatives) of the activation functions and hidden units, something Kolmogorov's representations don't have.
    - NN's are parameterised, the function Φ here is as complex to describe in bits as f itself.
    - However there is a version of Kolmogorov's ST that seems to be relevant for NNs https://direct.mit.edu/neco/article-pdf/3/4/617/812230/neco.1991.3.4.617.pdf
    - "It gives a possibility theorem but doesn't explain the performance"
- Given particular constraints on the hidden units or activation functions of a neural network its possible to get theoretical results that show these networks can express arbitrary functions but these aren't useful for general deep networks
- The paper uses the Stone-Weierstrass Theorem and the cosine squasher of Gallant and White to show that standard multilayer FF-NNs using arbitrary squashing functions.

Main results

Definition 2.2

- This sets out the class of output functions for single hidden layer NNs with squashing at the hidden layer and no squashing at the output layer
- Squashing function, following a affine transformation

Definition 2.3 - Squashing functions

- Squashing functions are functions which map $\mathbb{R} \to [0, 1]$ such that the function tends to 1 from below as the input $\to \infty$ and tends to 0 from above as the input $\to -\infty$
- A function is measurable if it preserves measurability, i.e if $f : X \to Y$, if $V \subset Y$ is measurable, then $f^{-1}(V) \subset X$ is measurable.
- These Sigma-Pi networks are a more complex generalisation of the NNs we know, I don't think they are widely used anymore due to exploding numbers of parameters.

Definition 2.5

- The class of output functions for single hidden layer NNs is Borel Measure Measurable(M^r), and if the activation function is continuous then it is continuous (C^r). $C^r \subset M^r$.
- The authors claim that almost all functions of interest in applications are continuous.

Definition 2.6

- Gives the definition of denseness, the $V \subset X$ is dense in $T \subset X$ if for every element t of T there exists an element of V which is ϵ close to t w.r.t some metric.
- In otherwords an element in V can approximate an element of T to an arbitrary degree of accuracy

Definition 2.7

- Compact is a generalisation of closed and bounded
- Closed as in it contains its endpoints (or limit points for seqeuences in the set)
- Uniform convergence means we don't need to know the x the functions are evaluated at, they are getting close on the whole domain.

Theorem 2.1

- The Sigma-Pi networks, with any continous nonconstant function as activation function, are uniformally dense on compact subsets of the continuous functions from $\mathbb{R}^r \to \mathbb{R}$.

Proof Sketch

- We want to apply the Stone-Weierstrass Theorem to show the sigma-pi networks are dense on compacta in C^r
- K an arbitrary compact set in $\mathbb{R}^r$
- For any continuous non-constant function g the set of sigma-pi is a subalgebra of $C(K, \mathbb{R})$
- Show it seperates points, using the fact for any points $x \neq y$ with $g(x) \neq g(y)$ we can find affine transformation s.t $A(a) = x, A(b) = y$
- Show there are constant, non-zero functions $g(A(\cdot))$, in a similar way
- Then stone-weierstrass applies.

Definition 2.8

- Two measurable functions are (measure) μ-equivalent if they only differ on a set of inputs of measure 0. (We can change the values of the function on this set without changing its integral, alternatively it has 'arbitrarily small volume')

Definition 2.9

- This defines a metric st.th two functions are close in metric if there is only a vanishing amount of measure on sets where they differ

Theorem 2.2

- "Single layer sigma-pi networks can approximate any measureable function"

Proof Sketch

- This proof use lemma A.1: Continuous functions are ρ_μ dense in the set of measurable functions
    - i.e we can approximate measurable functions with continuous functions
- Given a continuous nonconstant function (activation) it follows from Theorem 2.1 and lemma 2.2 that the sigma-pi networks using that "activation" are ρ_μ dense in the set of continuous functions. (Uniformly dense → sequences converge uniformly) Since the continuous functions are dense in the set of measureable functions the result follows after applying the triangle ineq.

Lemma A.2

- For a continous squashing function we can approximate this function with a sigma network using an arbitrary squashing function

Theorem 2.3

- For arbitrary squashing functions sigma-pi networks are uniformly dense on compact sets of C^r and dense in the measure sense on M^r

Proof Sketch

- From Theorem 2.2 and Lemma 2.2 we only need to show that $\Sigma\Pi^r(\Psi)$ is uniformly dense on $\Sigma\Pi^r(F)$ for some continuous squashing function F. (I.e need to show the uniformity condition for lemma 2.2 so theorem 2.2 applies showing we are close to measurable functions in the ρ_μ sense).
- So we need to show functions made up of products of F composed with affine transformations can be uniformly approximated by functions in $\Sigma\Pi^r(\Psi)$
- They show this Using lemma A.2

Theorem 2.4

- They then show that the simpler (and more interesting to us) sigma networks are universal approximators.
    - They do this by expressing the sigma-pi networks as trigonometric polynomials
    - This is possible because any continuous squashing function can be approximated by a sigma-pi network with an arbitrary squashing function (lemma A.2)
        1. Show we can approx $\cos(x)$ with a member of $\Sigma^r(\Psi)$
        2. Show we can approx weighted sum of $\cos(x)$ functions with a member of $\Sigma^r(\Psi)$
        3. Then use this to show $\Sigma^r(\Psi)$ is uniformly dense on compacta of C^r (Lemma A.5)
            1. Do this by noting that trigonometric polynomials $\sum \beta_i \prod_k \cos(A_i k(\cdot))$ are sigma pi networks and apply theorem 2.1
            2. The apply trig identity $\cos(a) \cdot \cos(b) = \cos(a+b) - \cos(a-b)$ to get to a sigma network
        4. Then apply Lemma A.5 + lemma 2.2 to show $\Sigma^r(\Psi)$ is ρ_μ dense in C^r, then triangle ineq to show ρ_μ dense in M^r

- After this they have some corollaries that extend these results to
    - L_p spaces
    - Boolean Functions
    - Functions with finite support
    - Multi-output networks
    - Multi-output, multi layer networks

Discussion and conclusion

- Have these results been extended to networks with inductive biases (RNNs, LSTMs, CNNs?)
    - I think they have been extended to GNNs
    - And multi-layer single output NNs
- These results show that functions of the form of a NN can approximate, they say nothing about the ability of SGD (for instance) to find functions that well approximate the function we want
- "We have thus established that such "mapping" networks are universal approximators. This implies that any lack of success in applications must arise from inadequate learning, insufficient numbers of hidden units or the lack of a deterministic relationship between input and target."
    - Is this true? We know we can have "too many" hidden units, or the network have hidden units distributed in a non-useful way (too wide)
- How to increase the number of hidden units as the amount of data increases
    - Is this relevant in the age of "scale"
    - "*I suspect that scaling up neural networks — these days, I don't hesitate to train a 20 million-plus parameter network (like ResNet-50) even if I have only 100 training examples — has also made them more robust*." (https://read.deeplearning.ai/the-batch/the-trouble-with-reinforcement-learning/)
- How to increase amount of data/number of hidden units as input space grows