GANNs: A New Theory of Representation for **Nonlinear** Bounded Operators

William Guss

Machine Learning at Berkeley

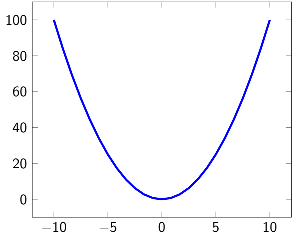
April 22, 2016

Introduction: What's up with continuous data?

- **All** of the data we deal with is discrete thanks to Turing.
- But, most of it models a continuous process.
- **Examples**
  - Audio: We take $> 100k$ samples of something we could describe with $f : \mathbb{R} \to \mathbb{R}$! Trick Question: Which is easier to use? (a) $v \in \mathbb{R}^{100000}$ or (b) f.
  - Images: We take $100k \times 100k$ samples of something we could describe with $f : \mathbb{R}^2 \to \mathbb{R}$.
- Why do we use discrete data? No computer known can really store f. End of stoy.

Introduction: Abusing continuity

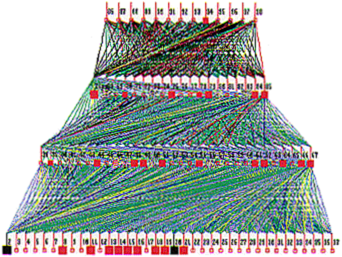
- f can't be *that* bad. Can it?
- If f is smooth it's easy to draw:



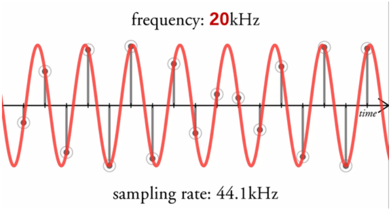
- I can even name f most of the time: $f : x \mapsto x^2$ or even super precisely $g : x \mapsto \sum_i^\infty a_n x^n$.
- Moral: Smooth functions are mostly very managable.

Introduction: Abusing continuity

- So why do we do this:



- To classify this:



The Core Idea: Let neural networks abuse continuity and smoothness.

Artificial Neural Networks

Definition

We say $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^m$ is a feed-forward neural network if for an input vector $\boldsymbol{x}$,

$$\begin{aligned} \mathcal{N} : \sigma_j^{(l+1)} &= g\left(\sum_{i \in Z^{(l)}} w_{ij}^{(l)} \sigma_i^{(l)} + \beta^{(l)}\right) \\ \sigma_i^{(0)} &= x_i \end{aligned}, \tag{1}$$

where $1 \leq l \leq L - 1$. Furthermore we say $\{\mathcal{N}\}$ is the set of all neural networks.

Operator Neural Networks

Let's get rid of $\mathbb{R}^{100000}$ and use f.

Definition

We call $\mathcal{O} : L^p(X) \to L^1(Y)$ an operator neural network if,

$$\begin{aligned} \mathcal{O} : \sigma^{(l+1)}(j) &= g\left(\int_{R^{(l)}} \sigma^{(l)}(i) w^{(l)}(i,j)\ di\right) \\ \sigma^{(0)}(j) &= f(j). \end{aligned} \tag{2}$$

Furthermore let $\{\mathcal{O}\}$ denote the set of all functional neural networks.

Well that was easy. In fact $\{\mathcal{O}\} \supset \{\mathcal{N}\}$
These definitions looks really similar? Is there some more general category or structure containing them.

Generalized Artifical Neural Networks

Definition

If A, B are (possibly distinct) Banach spaces over a field $\mathbb{F}$, we say $\mathcal{G} : A \to B$ is a generalized neural network if and only if

$$\begin{aligned} \mathcal{G} : \sigma^{(l+1)} &= g\left(T_l\left[\sigma^{(l)}\right] + \beta^{(l)}\right) \\ \sigma^{(0)} &= \xi \end{aligned} \tag{3}$$

for some input $\xi \in A$, and a linear form T_l.

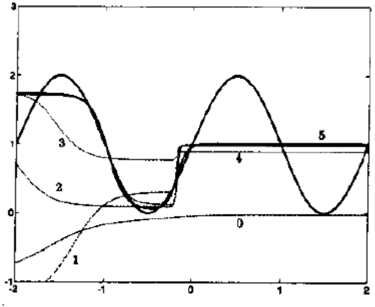
Claim: "Neural networks" are powerful because they can move bumps anywhere!
How? T_l is a linear form. It can move $\sigma^{(l)}$ anywhere, and g is a bump of some sort.

Moving bumps around

- The sigmoid function

$$g = \frac{1}{1 + e^{-x}} \tag{4}$$

is a bump, that we can move around with weights!



T_l as the layer type.

Definition

We suggest several classes of T_l as follows

- T_l is said to be $\mathfrak{o}$ operational if and only if $=$

$$\begin{aligned} T_l = \mathfrak{o} : L^p(R^{(l)}) &\to L^1(R^{(l+1)}) \\ \sigma &\mapsto \int_{R^{(l)}} \sigma(i) w^{(l)}(i,j)\ di. \end{aligned} \tag{5}$$

- T_l is said to be $\mathfrak{n}$ discrete if and only if

$$\begin{aligned} T_l = \mathfrak{n} : \mathbb{R}^n &\to \mathbb{R}^m \\ \vec{\sigma} &\mapsto \sum_j^m \vec{e}_j \sum_i^n \sigma_i w_{ij}^{(l)} \end{aligned} \tag{6}$$

where $\vec{e}_j$ denotes the j^{th} basis vector in $\mathbb{R}^m$.

T_l as the layer type.

Definition

- T_l is said to be $\mathfrak{n}_1$ transitional if and only if

$$\begin{aligned} T_l = \mathfrak{n}_1 : \mathbb{R}^n &\to L^q(R^{(l+1)}) \\ \vec{\sigma} &\mapsto \sum_i^n \sigma_i w_i^{(l)}(j). \end{aligned} \tag{7}$$

- T_l is said to be $\mathfrak{n}_2$ transitional if and only if

$$\begin{aligned} T_l = \mathfrak{n}_2 : L^p(R^{(l)}) &\to \mathbb{R}^m \\ \sigma(i) &\mapsto \sum_j^m \vec{e}_j \int_{R^{(l)}} \sigma(i) w_j^{(l)}(i)\ di \end{aligned} \tag{8}$$

Neural networks as diagrams!

This generalization is nice from a creative standpoint.
I can come up with new sorts of "classifiers" on the fly.
Examples:

- A three layer neural network is just

$$\mathcal{N}_3 : \mathbb{R}^{10000} \xrightarrow{g \circ \mathfrak{n}} \mathbb{R}^{30} \xrightarrow{g \circ \mathfrak{n}} \mathbb{R}^3. \tag{9}$$

- A three layer operator network is simply

$$\mathcal{O}_3 : L^p(R) \xrightarrow{g \circ \mathfrak{o}} L^1(R) \xrightarrow{g \circ \mathfrak{o}} C(R). \tag{10}$$

- We can even classify functions!

$$\mathcal{C} : L^p(R) \xrightarrow{g \circ \mathfrak{o}} L^1(R) \xrightarrow{g \circ \mathfrak{o}} \dots \xrightarrow{g \circ \mathfrak{o}} L^1(R) \xrightarrow{g \circ \mathfrak{n}_2} \mathbb{R}^n. \tag{11}$$

Results: Did abusing continuity help?

For every layer $\mathfrak{o}$ has weights

$$w^{(l)}(i,j) = \sum_{b}^{Z_Y^{(l)}} \sum_{a}^{Z_X^{(l)}} k_{a,b}^{(l)} i^a j^b. \tag{12}$$

Theorem

Let $\mathcal{C}$ be a GANN with only one $\mathfrak{n}_2$ transitional layer with $O(1)$ weight polynomial. If a continuous function, say $f(t)$ is sampled uniformly from $t = 0$, to $t = N$, such that $x_n = f(n)$, and if $\mathcal{G}$ has an input function which is piecewise linear with $O(N^2)$ weights.

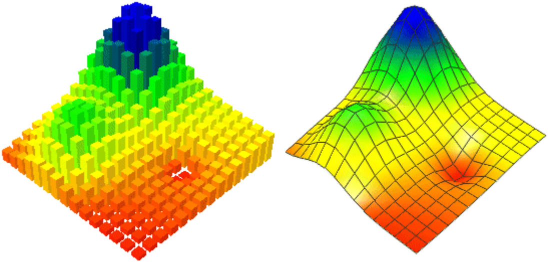
$$\xi = (x_{n+1} - x_n)(z - n) + x_n \tag{13}$$

for $n \leq z < n + 1$, then there exist some discrete neural network $\mathcal{N}$ such that $\mathcal{G}(\xi) = \mathcal{N}(\boldsymbol{x})$.

Results: Did abusing continuity help?

WHAT!?!? How did $\mathcal{C}$ reduce the number of weights from $O(N^2)$ to $O(1)$?

- The infinite dimensional versions of $\mathcal{N}$, in particular $\mathcal{O}$ and $\mathcal{C}$ are invariant to input quality. Takes the idea behind Convnets to an extreme!
- This is easy to see.



Results: Representation Theory

How good are Continuous Classifier Networks, $\{\mathcal{C}\}$ as algorithms?

Theorem

Let X be a compact Hausdorf space. For every $\epsilon > 0$ and every continuous bounded functional on $L^q(X)$, say f, there exists a two layer continuous classifier

$$\mathcal{C} : L^q(x) \xrightarrow{g \circ \mathfrak{n}_2} \mathbb{R}^m \xrightarrow{\mathfrak{n}} \mathbb{R}^n \tag{14}$$

such that

$$\|f - \mathcal{C}\| < \epsilon. \tag{15}$$

Results: Representation Theory

How good are Operator Networks and GANNs as algorithms? They should be able to approximate the important operators, eg. **Fourier Transform, Laplace Transform**, **Derivation**, etc.

Theorem

Given a operator neural network $\mathcal{O}$ then some layer $l \in \mathcal{O}$, the let $K : C(R^{(l)}) \to C(R^{(l)})$ be a bounded linear operator. If we denote the operation of layer l on layer $l-1$ as $\sigma^{(l+1)} = g\left(\Sigma_{l+1}\sigma^{(l)}\right)$, then for every $\epsilon > 0$, there exists a weight polynomial $w^{(l)}(i,j)$ such that the supremum norm over $R^{(l)}$

$$\left\| K\sigma^{(l)} - \Sigma_{l+1}\sigma^{(l)} \right\|_\infty < \epsilon \qquad (16)$$

Proof.

See paper. Nice! □

Results: Stronger Representation Theory

We want to show the following better theorem.

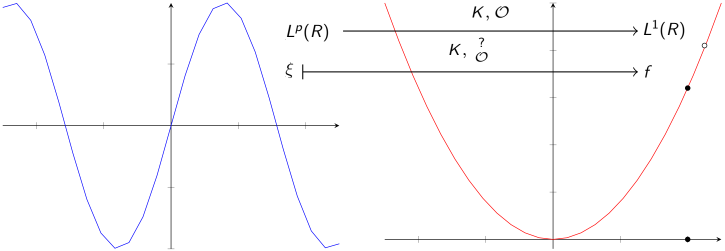
Theorem

Given a operator neural network $\mathcal{O}$ then some layer $l \in \mathcal{O}$, the let $K : C(R^{(l)}) \to C(R^{(l)})$ be a bounded **continuous** *operator. If we denote the operation of layer l on layer $l-1$ as $\sigma^{(l+1)} = g\left(\Sigma_{l+1}\sigma^{(l)}\right)$, then for every $\epsilon > 0$, there exists a weight polynomial $w^{(l)}(i,j)$ such that the supremum norm over $R^{(l)}$*

$$\left\| K\sigma^{(l)} - \Sigma_{l+1}\sigma^{(l)} \right\|_\infty < \epsilon \tag{17}$$

But how? **Dirac Spikes!**

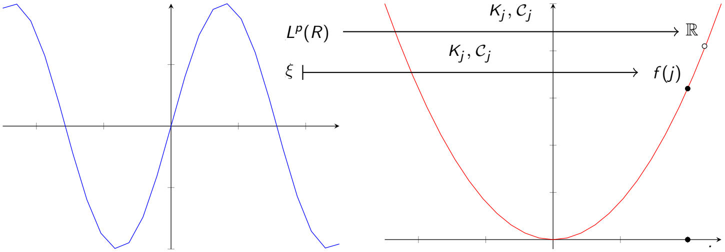
Results: Stronger Representation Theory



Proof.

∎

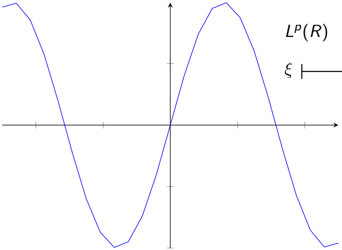
Results: Stronger Representation Theory

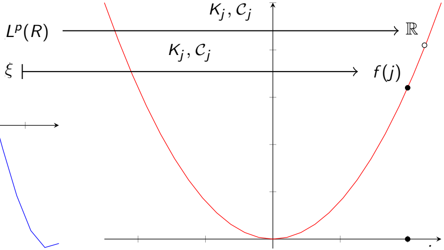


Proof.

- Fix $\epsilon > 0$. Given $K : \xi \mapsto f$, let $K_j : \xi \mapsto f(j)$ be a functional on L^q.

Results: Stronger Representation Theory



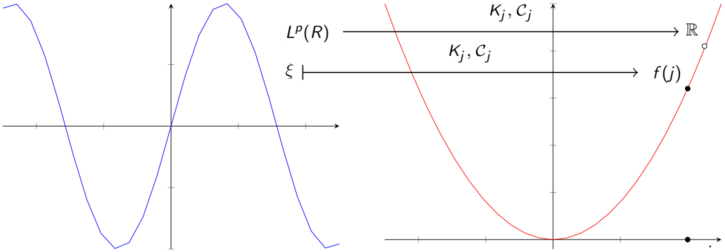


Proof.

- Fix $\epsilon > 0$. Given $K : \xi \mapsto f$, let $K_j : \xi \mapsto f(j)$ be a functional on L^q.
- We can find a $\mathcal{C}_j : L^q(R) \xrightarrow{g \circ \mathfrak{n}_2} \mathbb{R}^{m(j)} \xrightarrow{\mathfrak{n}} \mathbb{R}^1$ so that for all ξ,

$$|\mathcal{C}_j(\xi) - K_j(\xi)| = |\mathcal{C}_j(\xi) - f(j)| < \epsilon/2. \qquad (18)$$

Results: Stronger Representation Theory



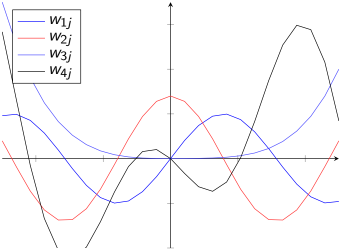
Proof.

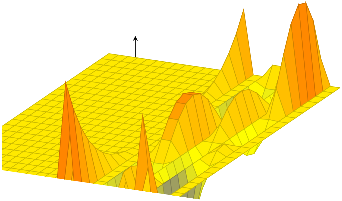
- We know that

$$\mathcal{C}_j(\xi) = \sum_{k=1}^{m(j)} a_{jk} g\left(\int_R \xi(i) w_{kj}(i)\ d\mu(i)\right) \tag{18}$$

□

Results: Stronger Representation Theory





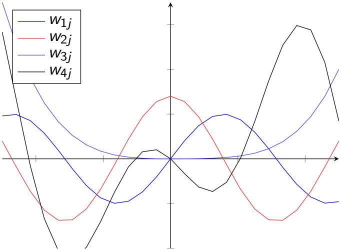
Proof.

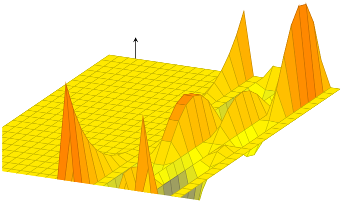
- We wish to turn C_j into a two layer $\mathcal{O}$. Let,

$$w^{(0)}(i,\ell) = \begin{cases} w_{kj}(i), & \text{if } \ell = j + k,\ \text{k} \in 1, \ldots, m(j) \\ 0 & \text{otherwise} \end{cases}$$

$\square$

Results: Stronger Representation Theory





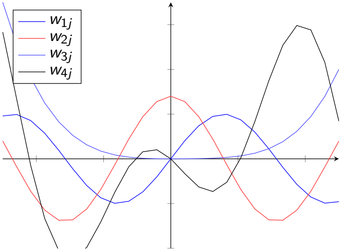
Proof.

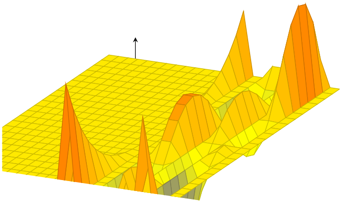
- Then

$$\mathcal{C}_j(\xi) = \sum_{k=1}^{m} a_{jk} g \circ \mathfrak{o}[\xi](k+j) \tag{18}$$

□

Results: Stronger Representation Theory





Proof.

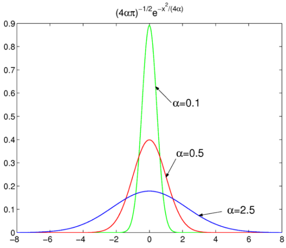
- Then

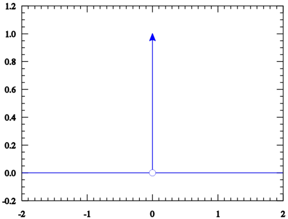
$$\mathcal{C}_j(\xi) = \sum_{k=1}^{m} a_{jk} g \circ \mathfrak{o}[\xi](k+j) \tag{18}$$

- How do we turn this finite sum into an integral? Dirac time!

$\square$

Results: Stronger Representation Theory





Proof.

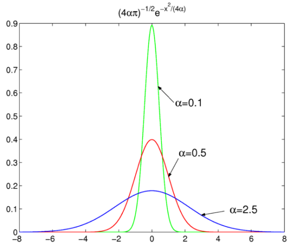
- We define a dirac spike as follows for every n:

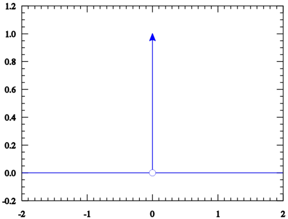
$$\delta_{nkj}(\ell) = cn\exp(-bn^2|\ell - (j+k)|^2) \tag{18}$$

where c, b are set so that $\int_{\mathbb{R}} \delta_{nkj} = 1$

□

Results: Stronger Representation Theory





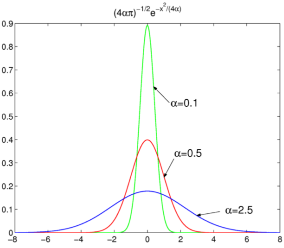
Proof.

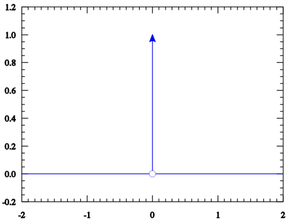
- Now let the second weight function be:

$$w_n^{(1)}(\ell, j) = \sum_{k=1}^{m} a_{jk}\delta_{nkj}(\ell) \tag{18}$$

□

Results: Stronger Representation Theory





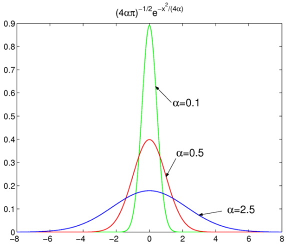
Proof.

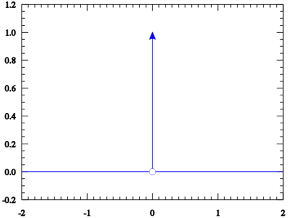
- Putting everything together, for every n let $\mathcal{O}_n : L^p(R) \to L^1([0,1])$

$$\mathcal{O}_n : \xi \mapsto \int_R w^{(1)}(\ell, j)\mathfrak{o}[\xi](\ell)\ d\mu(\ell). \tag{18}$$

Clearly $\mathcal{O}_n \to \sum_{k=1}^m a_{jk} g \circ \mathfrak{o}[\xi](k+j)$

Results: Stronger Representation Theory





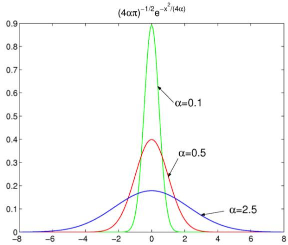
Proof.

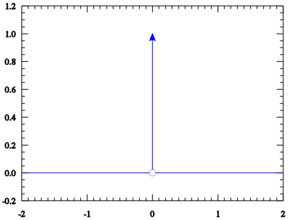
- Therefore for every $\epsilon > 0$ there exists an N such that for all $n > N$, for all ξ, and for all j,

$$|O_n[\xi](j) - C_j[\xi]| \leq \|\mathcal{O}_n[\cdot](j) - C_j[\cdot]\| < \epsilon/2. \tag{18}$$

□

Results: Stronger Representation Theory





Proof.

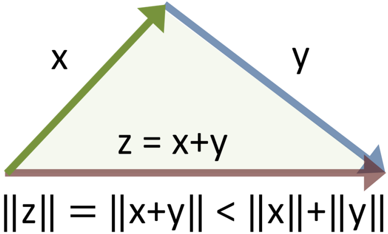
- Therefore for every $\epsilon > 0$ there exists an N such that for all $n > N$, for all ξ, and for all j,

$$|O_n[\xi](j) - C_j[\xi]| \leq \|\mathcal{O}_n[\cdot](j) - C_j[\cdot]\| < \epsilon/2. \tag{18}$$

- Recall that for every j, $\|K_j - C_j\| < \epsilon/2$.

$\square$

Results: Stronger Representation Theory



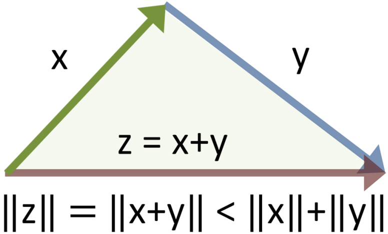
Proof.

- By the triangle inequality we have that for all j

$$\begin{aligned} \|K_j - \mathcal{O}_n(k)\| &= \|K_j - \mathcal{O}_n(j) + C_j - C_j\| \\ &\leq \|K_j - C_j\| + \|\mathcal{O}_n(j) - C_k\| < \epsilon. \end{aligned} \tag{18}$$

□

Results: Stronger Representation Theory



Proof.

- By the triangle inequality we have that for all j

$$\begin{aligned} \|K_j - \mathcal{O}_n(k)\| &= \|K_j - \mathcal{O}_n(j) + C_j - C_j\| \\ &\leq \|K_j - C_j\| + \|\mathcal{O}_n(j) - C_k\| < \epsilon. \end{aligned} \tag{18}$$

- Therefore $\|K - \mathcal{O}\| < \epsilon$

$\square$