Dimensionality Reduction Using Generalized Artificial Neural Networks

William Guss WGUSS@BERKELEY.EDU
Machine Learning at Berkeley, 246 Cory Hall, Berkeley, CA 94720 USA

Abstract

In this paper, we generalize ANNs to infinite dimensional Banach spaces by developing a practical analog to the feedforward propagation algorithm. Using this new class of algorithms, GANN, we prove a new universal approximation theorem for bounded linear operators and show that representation of weights by samples from a multivariate weight polynomial can drastically reduce the dimensionality of a learning problem. Lastly, we give a practical implementation of the error back-propagation algorithm in this space for the classification of continuous data.

1. Introduction

Although neural networks have proven an extremely effective mechanism of machine learning (Burch, 2001), theoretically they remain a black-box model. In answer to this problem (Neal, 2012) examined the notion of infinite hidden nodes with a network proving that such a construction becomes a Gaussian kernel. Then, (Roux & Bengio, 2007) described a model for affine neural networks with continuous hidden layers in alignment with (Neal, 2012). These authors showed effectively the viability of a "continuous" neural network, but left many similar constructions unexplored.

It is the subject of this paper to generalize the construction of a feed-forward ANN which maps uncountably infinite vector spaces, and then to demonstrate the practical implementations of algorithms such as error backpropagation in this generalized form.

2. Functional Neural Networks

In the standard feed-forward case proposed in (McCulloch & Pitts, 1943), if on the lth layer, $Z^{(l)} = 1, \dots, n$ is the index set of neurons, β is the bias, g is the sigmoid activation function, and σ_j is the output of the jth neuron, the following recurrence relation is natural.

Definition 1. *We say $\mathcal{N} : \mathbb{R}^n \to \mathbb{R}^m$ is a feed-forward neural network if for an input vector $\boldsymbol{x}$,*

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

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

Suppose that we wish to map one functional space to another with a neural network. Consider the standard model of an ANN as the number of neural nodes for every layer becomes uncountable. The index for each node then becomes real-valued, along with the weight and input vectors.

Let us denote $\xi : X \subset \mathbb{R} \to \mathbb{R}$ as some arbitrary input function for the neural network. Likewise consider a real-valued weight function, $w^{(l)} : \mathbb{R}^2 \to \mathbb{R}$, for a layer l which is composed of two indexing variables $i, j \in \mathbb{R}$. Finally as the number of neural nodes becomes uncountable we define a real-valued bound for any given layer $R^{(l)} \supset Z^{(l)}$. As the indices become real valued, examination of the weighted sum seen in Definition 1 leads us to the following abuse of notation which is not too dissimilar from the derivation of the Laplace Transform from the power series.

$$\begin{aligned} \sigma^{(1)}(j) &= \lim_{\Delta i \to 0} g\left(\sum_{i \in R^{(0)}} \xi(i) w^{(0)}(i,j)\, \Delta i + \beta\right) \\ &= g\left(\int_{R^{(0)}} \xi(i) w^{(0)}(i,j)\, di\right) \end{aligned} \tag{2}$$

The β bias term is omitted as it can surely be formed as an artifact of integration of the weight function. Repeating the process inductively for all layers in a neural network, leads to a recurrence relation for this construction.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

Definition 2. *We call $\mathcal{F} : L^1(X) \to L^1(Y)$ a functional neural network if,*

$$\begin{aligned} \mathcal{F} : \sigma^{(l+1)}(j) &= g\left(\int_{R^{(l)}} \sigma^{(l)}(i) w^{(l)}(i,j)\, di\right) \\ \sigma^{(0)}(j) &= \xi(j). \end{aligned} \tag{3}$$

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

To demonstrate the accuracy of this generalization, let us propose the following theorem which is near that of (Roux & Bengio, 2007).

Theorem 3. *Suppose $\mathcal{F}$ is a functional neural network with a set of piecewise constant weight functions $W = \{w^{(0)}, w^{(1)}, \ldots, w^{(L-1)}\}$ each with constituent pieces of length one. Then given the input function $\xi(i) = x_n, n = \max\{m \in \mathbb{Z} \mid m \leq i\}$ for some input vector $\boldsymbol{x}$, $\mathcal{F}$ is discretized; that is assuming $i, j \in \mathbb{R}$ and $Z^{(l)} = R^{(l)} \cap \mathbb{Z}$ then for $0 \leq l \leq L-1$ we have that*

$$\mathcal{F}(\xi) = \mathcal{N}(\boldsymbol{x}), \tag{4}$$

so $\{\mathcal{F}\} \supset \{\mathcal{N}\}$

Proof. Let $P(l)$ be the proposition that $\sigma^{(l+1)}(j)$ becomes discretized when $w^{(l)}(i,j)$ and $\sigma^{(l)}(i,j)$ are piecewise constant with constituent functions of length one. Moreover let $w_{ij}^{(l)}$ denote the value of $w^{(l)}(i,j)$ for $(i,j) = \max\{(x,y) \in \mathbb{Z}^2 \mid x \leq i \wedge y \leq j\}$. Then by induction we show $P(l), 0 \leq l \leq L-1$.

Basis Step. Recall that $\sigma^{(0)}(j) = \xi(j)$. Then one would suppose

$$\sigma^{(1)}(j) = g\left(\int_{R^{(0)}} \xi(i) w^{(0)}(i,j)\, di+\right) \tag{5}$$

but because the weight function and the input function are piecewise constant and not guaranteed to be continuous for $R^{(0)}$, we must take the improper integral along each constituent piece of length one. Supposing that each summation in the following is taken over $k \in Z^{(0)}$,

$$\begin{aligned} \sigma^{(1)}(j) &= g\left(\sum_k \lim_{b\to k} \int_{k-1}^{b} \xi(i) w^{(0)}(i,j)\, di+\right) \\ &= g\left(\sum_k w_{kj}^{(0)} \lim_{b\to k} \int_{k-1}^{b} \xi(i)\, di+\right) \\ &= g\left(\sum_k w_{kj}^{(0)} x_b+\right) \end{aligned} \tag{6}$$

Inductive Step. Now assume that for some l we have that

$$\sigma^{(l+1)}(j) = g\left(\sum_{i \in Z^{(l)}} w_{ij}^{(l)} \sigma_i^{(l)}+\right) \tag{7}$$

We now show that by the inductive hypothesis if $P(0) \wedge P(l) \to P(l+1)$, then $P(k)\ \forall k$. Consider the next neural layer defined as

$$\sigma^{(l+2)}(j) = g\left(\int_{R^{(l+1)}} \sigma^{(l+1)}(i) w(i,j)\, di+\right) \tag{8}$$

Then because $w(i,j)$ and $\sigma^{(l+1)}$ are piecewise constant by definition and not necessarily continuous for $R^{(l+1)}$ we must again take the improper Riemann integral over the constituent pieces. Consider $k \in Z^{(l+1)}$

$$\begin{aligned} \sigma^{(l+2)}(j) &= g\left(\sum_k \lim_{b\to k} \int_{k-1}^{b} \sigma^{(l+1)}(i) w(i,j)\, di+\right) \\ &= g\left(\sum_k w_{kj}^{(l+1)} \sigma_k^{(l+1)} \lim_{b\to k} \int_{k-1}^{b} di+\right) \\ &= g\left(\sum_k w_{kj}^{(l+1)} \sigma_k^{(l+1)}+\right) \end{aligned} \tag{9}$$

Therefore by the inductive hypothesis the proof follows and $\mathcal{F}(\xi) = \mathcal{N}(\boldsymbol{x})$ for piecewise constant input and weight functions.

□

From the logic of the preceding proof we can establish that the input function need only be properly Lebesgue integrable over $R^{(0)}$. Moreover, we come to an extremely important intuition; the weight matrix for a given layer l can be thought of as a piecewise constant weight surface, and the linear combination of weights can be thought of as a piecewise integral transformation along a given j on the weight surface. However, functional neural networks allow for infinite weight surfaces and therefore can represent the entire class of integral transforms. With this in mind, it is now possible to consider functional neural networks as universal approximators.

3. Universal Approximation of Bounded Linear Operators

In the case of discretized neural networks, George Cybenko and Kolmogorov have shown that with sufficient weights and connections, a feed-forward neural network is a universal approximator of arbitrary $f : \mathbb{R}^n \to \mathbb{R}^m$ (Cybenko, 1989); that is, constructs of the form $\mathcal{N}(\boldsymbol{x})$ are dense in $C(I^n, \mathbb{R}^m)$ where I^n is the unit hypercube $[0,1]^n$. Cybenko proved this remarkable result by utilizing the Riesz Representation Theorem for Hilbert spaces and the Hahn-Banach theorem. He showed by contradiction that there exists no bounded linear functional $h(x)$ in the form of $\mathcal{N}(\boldsymbol{x})$ such that $\int_{I_n} h(x)\, d\mu(x) = 0$.

Although this is result legitimizes neural networks, it is rather vacuous since it does not actually impart constraints on the type of networks which might approximate these functions. Fortunately using the intuitions presented in Theorem 1, it would be advantageous to examine the generalization of Cybenko's theorem to the larger class of functional neural networks. However, there is no clear way with which to do this; discretized neural networks map vector spaces and therefore approximate continuous functions, whereas functional neural networks are defined as arbitrary mappings between Hilbert spaces (more specifically the set of L^2 integrable functions). Moreover letting n approach infinity in Cybenko's proof fails to hold in that there is not an obvious topology for $C(C(\mathbb{R}), C(\mathbb{R}))$ which satisfies it. Therefore we must develop an approximation theorem for $\mathcal{F} : C(X) \to C(Y)$ over the set of linear operators.

First, however, let us develop the notion of $\mathcal{F}$ as a universal approximator of arbitrary functions. By Theorem 1, we have that $\{\mathcal{F}\} \supset \{\mathcal{N}\}$,and in that sense for piecewise constant $w(i,j)$ and $\xi(i)$ functional neural networks approximate any arbitrary mapping from $\mathbb{R}^n \to \mathbb{R}$ where $n = |Z^{(0)}|, m = |Z^{(L-1)}|$ by Cybenko's theorem. However, when considering the fully continuous case the following corollary arises from the Stone-Weierstrass theorem.

Corollary 4. *Suppose $\mathcal{F}$ is a multi-layer functional ANN. Then for some real-valued continuous function $f : \mathbb{R} \to \mathbb{R}$, there exists a set of weights W such that $\forall \epsilon > 0$, $\|\mathcal{F}(\xi) - f\|_\infty < \epsilon$*

Proof. In this proof we will omit the inductive step as it is elementary and employs the same logic as the basis step. Consider the first neural layer

$$\sigma^{(1)}(j) = g\left(\int_{R^{(0)}} \xi(i) w^{(0)}(i,j)\, di\right) \tag{10}$$

because we take $w^{(0)}$ to be some approximating polynomial by the Stone-Weierstrass theorem, let $w^{(0)} = \left[(g^{-1})' \circ (h(\Xi, j))\right] h'(\Xi, j)$ approximately, where Ξ is the primitive of ξ. Supposing that h is some polynomial satisfying $h(\Xi, j)\Big|_{R^{(0)}} = f(j)$, then by the chain rule of integration

$$\begin{aligned}\sigma^{(1)}(j) &= g\left(\int_{R^{(0)}} \xi(i) \left[(g^{-1})' \circ (h(\Xi, j))\right] h'(\Xi, j)\, di\right) \\ &= g\left(g^{-1}\left(h(\Xi, j)\right)\right)\Big|_{R^{(0)}} \\ &= f(j)\end{aligned} \tag{11}$$

□

At this point it is important to note that the proof given above implies that ξ is disregarded through manipulation of $w^{(0)}$. Instead, h is a function of Ξ which is the primitive of ξ. If we were to not let $h(\Xi, j)\Big|_{R^{(0)}} = f(j)$, then we could claim that for arbitrary h we have proven any functional composition of ξ is possible; that is, in some light sense we have proven Cybenko's theorem in the general case by treating the weight set as some hypersurface. Moreover, the intuition follows that if we were to discretized the satisfying h and ξ (Theorem 1) then it is possible that a similar weight surface is developed for a trained $\mathcal{N}$. This result is remarkable as new light is shed on the black-box model of neural networks showing that approximation of h is made in the discrete sense.

Although we have shown through the corollary that approximation of arbitrary functional composition is possible, we have yet to consider values of w in the general sense. In other words, what can be said about the general approximation of bounded linear operators mapping $C(\mathbb{R}^n)$ to $C(\mathbb{R}^n)$ where C denotes the set of continuous (integrable) real valued functions? Evidently, the form of $\mathcal{F}$ resembles the general class of integral transforms, $\int f(x) \cdot E(x,k)\, dx$. Integral transforms are shown to approximate a multitude of operators through varying kernel functions. For example consider some $g(t)$ and the integral transform

$$(\mathcal{P}g)(s) = \int_0^\infty g(t)\delta'(s-t)\, dt \tag{12}$$

where δ is the Dirac-Delta function. Then we have that $(\mathcal{P}g)(s) = \frac{dg}{dt}\Big|_s$ by the properties of the Delta function. Similar approximations of linear operators can be made by varying the kernel function E. Thus there is considerable interest in determining the density of integral transforms and thereby functional neural networks in the set of bounded linear operators.

Further investigation into dense forms of bounded linear operators leads us to the Schwartz theorem of bounded linear operator representation by integral kernels. The theorem simply states that all linear operators can be represented in some light sense by integral transforms with arbitrary kernels. However, this theorem is too general for our purposes and we would like to show that in the specific case of some functional neural network $\mathcal{F}$ that for any given layer such that $l \neq 0$ any linear operator can be approximated with point-wise convergence from the Weierstrass polynomial approximation theory.

In order to do this we will return to the Riesz representation theorem that states the following (Hartig, 1983).

Theorem 5. *Let $\phi : C(X) \to \mathbb{R}$ be any bounded linear form where X is a compact Hausdorff space and $C(X)$ is the Banach space of continuous functions over X. Then*

there exists a unique regular Borel measure μ on X such that

$$\phi(f) = \int_X f(t)\, d\mu(t), \quad f \in C(X), \quad t \in X \tag{13}$$

and $\|\phi\| = |\mu|(X)$ where $|\mu|$ is the total variation of μ on X.

As opposed to generalizing Cybenko's theorem to Banach spaces ($\mathbb{R}^\infty$), we can actually manipulate the representation theorem to encapsulate bounded linear operators over locally compact Hausdorf spaces. Using the aforementioned logic the universal representation theorem for functional neural networks is now proposed.

Theorem 6. *Given a functional neural network $\mathcal{F}$ then some layer $l \in \mathcal{F}$, 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 $R^{(l)}$*

$$\left\| K\sigma^{(l)} - \Sigma_{l+1}\sigma^{(l)} \right\|_\infty < \epsilon \tag{14}$$

Proof. Let $\zeta_t : C(R^{(l)}) \to R^{(l)}$ be a linear form which evaluates its arguments at $t \in R^{(l)}$; that is, $\zeta_t(f) = f(t)$. Then because $\zeta_t$is bounded on its domain, $\zeta_t \circ K = K^\star\zeta_t$ is a bounded linear functional. Then from the Riesz Representation Theorem (Theorem 1) we have that there is a unique regular Borel measure μ_t on $R^{(l)}$ such that

$$\left(K\sigma^{(l)}\right)(t) = K^\star\zeta_t\left(\sigma^{(l)}\right) = \int_{R^{(l)}} \sigma^{(l)}(s)\, d\mu_t(s), \quad \|\mu_t\| = \|K^\star\zeta_t\| \tag{15}$$

Then if there exists a regular Borel measure μ such that μ_t is significantly smaller that μ for all t, then we have that, by the Radon-Nikodim derivative, $d\mu_t(s) = K_t(s)d\mu(s)$ under the assumption that K_t is L^1 integrable over $R^{(l)}$ with the measure μ. Thus it follows that

$$\begin{aligned} K\left[\sigma^{(l)}\right](t) &= \int_{R^{(l)}} \sigma^{(l)}(s)K_t(s)\, d\mu(s) \\ &= \int_{R^{(l)}} \sigma^{(l)}(s)K(t,s)\, d\mu(s). \end{aligned} \tag{16}$$

Therefore, for any bounded linear operator $K : C(X) \to C(X)$ there exists a unique $K(t,s)$ such that $K[f] = \int_X f(s)K(t,s)d\mu(s)$. Now we show that the operation of Σ_l can approximate any such operator. Because K is of the form of Σ_l where the only difference is the weighting function, we assert the following claim.

Let G be defined as a linear functional applied to a Gaussian heat kernel whose application with a function $f : \mathbb{R} \to \mathbb{R}$ yields the following definition,

$$G[f](x) = \frac{1}{b\sqrt{\pi}} \int_{\mathbb{R}} f(u)\mathrm{e}^{\left(\frac{u-x}{b}\right)}\, du. \tag{17}$$

Then it follows that by the Weierstrass approximation theorem that for all $\epsilon > 0$, the supremum norm $\|f - G[f]\|_\infty < \epsilon$. Then because G is a polynomial, f must be a limit of polynomials. So now consider the operation of $K\left[\sigma^{(l)}\right](t)$ with kernel $K(t,s)$. By the Weierstrass approximation theorem $K(t,s)$ must be a limit of polynomials and therefore we let $w^{(l)}(i,j)$ assume that limit. That is,

$$\begin{aligned} \lim_{b\to 0} \left\| K\left[\sigma^{(l)}\right] - \int_{R^{(l)}} \sigma^{(l)}(s)\, G[k]\, d\mu(s) \right\|_\infty = \\ \left\| K\left[\sigma^{(l)}\right] - \Sigma_{l+1}\left[\sigma^{(l)}\right] \right\|_\infty < \epsilon \end{aligned} \tag{18}$$

Thus we have that as a limit of polynomials the operation of any arbitrary layer of a functional neural network $\mathcal{F}$ approaches any arbitrary linear bounded operator $K : C(R^{(l)}) \to C(R^{(l)})$; that is, functionals of the form Σ_{l+1} are dense in the set of all bounded continuous linear operators. □

In the above proof, a remarkable fact has been shown: functional neural networks can perform most mathematical operations on their inputs, and so neural networks preserve universal approximation in infinite dimensions!

4. Generalized Artificial Neural Networks

To implement Functional Neural Networks in a meaningful context, we need some algorithm which uses the computational power of $\{\mathcal{F}\}$ to yield finite dimensional classifications of ξ.

So the question arises: *does there exist some more general structure which includes FNNs and ANNs without piecewise approximation?* The solution to such questions must furthermore maintain universal approximation and computational evaluability so as to allow the implementation of a real-world algorithm.

Therefore, in this section of the paper we will propose the Generalized Artificial Neural Network through the examination of specific operations on different layers and then search the space of these GANNs for an algorithm that lets us make use of $\{\mathcal{F}\}$.

4.1. The Generalization of ANNs

In order to generalize ANNs in a way that follows logically, we take the initial definition of functional neural networks and consider the notion of a layer.

Definition 7. *If A, B are (possibly distinct) Banach spaces*

over $\mathbb{R}$, we say $\mathcal{G} : A \to B$ is a generalized neural network if and only if

$$\begin{aligned} \mathcal{G} : \sigma^{(l+1)} &= g\left(T_l\left[\sigma^{(l)}\right] + \beta^{(l)}\right) \\ \sigma^{(0)} &= \xi \end{aligned} \tag{19}$$

for some input $\xi \in A$..

In (19), it is unclear how the form of T_l is restricted. It might be recalled that T_l takes a form similar to the operation of a layer $l+1$ on l in the proof of universal approximation for $\{\mathcal{F}\}$. Let us consider a few other such forms of T_l by first stating a formal definition.

Definition 8. *We say that T_l is the operation of a layer $l+1$ on l in some $\mathcal{G}$ if and only if for $l = L-1$, $T : C \to B$ with C the codomain of $\sigma^{(0)}$ and for $l = 0$, $T_l : A \to D$ where D is the domain of $\sigma^{(1)}$.*

Using the above definition it is now possible to construct different classes of T_l using ideas directly from the constructions of $\mathcal{N}$ and $\mathcal{F}$.

Definition 9. *We suggest several classes of T_l as follows*

- *T_l is said to be $\mathfrak{f}$ functional if and only if =*

$$\begin{aligned} T_l = \mathfrak{f} : C(R^{(l)}) &\to C(R^{(l+1)}) \\ \sigma &\mapsto \int_{R^{(l)}} \sigma(i) w^{(l)}(i,j)\, di. \end{aligned} \tag{20}$$

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

$$\begin{aligned} T_l = \mathfrak{n} : \mathbb{R}^n &\to \mathbb{R}^m \\ \vec{\sigma} &\mapsto \sum_j^m \vec{e}_j \sum_i^n \sigma_i w_{ij}^{(l)} \end{aligned} \tag{21}$$

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

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

$$\begin{aligned} T_l = \mathfrak{n}_1 : \mathbb{R}^n &\to C(R^{(l+1)}) \\ \vec{\sigma} &\mapsto \sum_i^n \sigma_i w_i^{(l)}(j). \end{aligned} \tag{22}$$

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

$$\begin{aligned} T_l = \mathfrak{n}_2 : C(R^{(l)}) &\to \mathbb{R}^m \\ \sigma(i) &\mapsto \sum_j^m \vec{e}_j \int_{R^{(l)}} \sigma(i) w_j^{(l)}(i)\, di \end{aligned} \tag{23}$$

Remark 10. *Observe that without loss of generality $\{G\} \supset \{F\} \cup \{N\}$; that is, every $\mathcal{N}$ can be expressed in terms of $\mathcal{G} : \mathfrak{n} \to \cdots \to \mathfrak{n}$, and the same for every $\mathcal{F}$.*

4.2. Continuous Classifiers and Dimensionality Reduction

In the case that data is sampled from a continuous process over time, it is general practice to build a feature vector in dimensions identically equal to the number of samples n. Then with such a feature vector, a practitioner can build a network architecture to accommodate her learning task. Unfortunately, if the input is of high quality, this introduces high-dimensionality into the weight matrix and thereby any optimization algorithm chosen.

However, searching $\{\mathcal{G}\}$ space yields a theoretically better approach.

Definition 11. *$\mathcal{G}$ is said to be a continuous classifier network if it is defined such that*

$$\mathcal{G} : \mathfrak{f} \to \mathfrak{f} \to \ldots \mathfrak{f} \to \mathfrak{n}_2, \tag{24}$$

and for every $l < L-1$,

$$w^{(l)}(i,j) = \sum_b^{Z_Y^{(l)}} \sum_a^{Z_X^{(l)}} k_{a,b}^{(l)} i^a j^b. \tag{25}$$

In construction every single weight polynomial can capture lower-order properties of the weight surface with even less dimensionality than a typical piecewise weight surface $(w_i j)$. In fact polynomials can best-fit a function of n discrete points in the worst case with n coefficients, but they typically perform better within some ϵ error. [Numerical Methods of Curve Fitting. By P. G. Guest, Philip George Guest. Page 349.] The key here is that the resolution of the input function ξ only affects the feed-forward step on the first $\mathfrak{f}$ layer, and not the number, n, of parameters $k_{a,b}$ in the model.

Suppose in some instance we have a weight polynomial on an $\mathfrak{n}_2$ with $M \in O(1)$ parameters, which has learned a model. Then the following theorem gives a direct comparison of an ANN performing the same task.

Theorem 12. *Let $\mathcal{G}$ be a GANN with only one $\mathfrak{n}_2$ transitional layer. 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*

$$\xi = (x_{n+1} - x_n)(z - n) + x_n \tag{26}$$

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

Proof. Recall that for the jth output neuron of a single layer discretized neural network,

$$\mathcal{N}_j(\boldsymbol{x}) = g\left(\sum_{i=1}^N w_{i,j} x_i + \beta\right). \tag{27}$$

Let this $\mathcal{N}$ compose the same sigmoid as the aforementioned $\mathfrak{n}_2$ transitional layer. We need only show that equivalence holds on the inside.

Because the definition of a network at the jth componet gives us

$$\mathcal{G}_j(\xi) = g\left(\mathfrak{n}_2[\xi] + \beta\right), \tag{28}$$

it follows that,

$$\begin{aligned}
\mathfrak{n}_2(\xi) &= \int_R \xi(t)\, u_j(t)\, dt \\
&= \sum_n^{N-1} \int_n^{n+1} \left((x_{n+1} - x_n)(z-n) + x_n\right) u_j(z)\, dz \\
&= \sum_n^{N-1} (x_{n+1} - x_n) \\
&\qquad \times \int_n^{n+1} (z-n) \sum_m^M k_{m,j} z^m\, dz \\
&\qquad + x_n \int_n^{n+1} u_j(z)\, dz \\
&= \sum_n^{N-1} (x_{n+1} - x_n) \\
&\qquad \times \sum_m^M k_{m,j} \left[\frac{1}{m+2} z^{m+2} - \frac{n z^{m+1}}{m+1}\right]_n^{n+1} \\
&\qquad + \sum_m^M k_{m,j} x_n \frac{1}{m+1} z^{m+1}\Big|_n^{n+1}
\end{aligned} \tag{29}$$

Now, let $V_{n,j} = \sum_m k_{m,j} \left[\frac{1}{m+2} z^{m+2} - \frac{n z^{m+1}}{m+1}\right]_n^{n+1}$ and $Q_{n,j} = \sum_m k_{m,j} \frac{1}{m+1} z^{m+1}\big|_n^{n+1}$. We can now easily simplify (29) using the telescoping trick of summation.

$$\begin{aligned}
\mathfrak{n}_2(\xi) &= x_N V_{N-1,j} \\
&\quad + \sum_{n=2}^{N-1} x_n \left(Q_{n,j} - V_{n,j} + V_{n-1,j}\right) \\
&\quad + x_1 \left(Q_{1,j} - V1, j\right).
\end{aligned} \tag{30}$$

By simply letting $w_{1,j} = \left(Q_{1,j} - V_{1,j}\right), w_{n,j} = \left(Q_{n,j} - V_{n,j} + V_{n-1,j}\right)$, and $w_{N,j} = V_{N-1,j}$ the following relation is satisfied, $\mathfrak{n}_2(\xi) = \mathfrak{n}(\boldsymbol{x})$. Hence, $\mathcal{G}(\xi) = \mathcal{N}(\boldsymbol{x})$ and the proof is complete. □

Notice that in the last proof, we went from $M \in O(1)$ dimensions to $O(N)$ where N is the number of samples on the input signal ξ! So there is constant space dimensionality reduction when using the $\mathfrak{n}_2$, and it is not to hard to see this for $\mathfrak{f}$.

Algorithm 1 Feedforward Propagation on $\{\mathcal{F}\}$

Input: input function ξ
for $l \in \{0, \ldots, L-1\}$ **do**
  for $t \in Z_X^{(l)}$ **do**
    Calculate $I_t^{(l)} = \int_{R^{(l)}} \sigma^{(l)}(j_l) j_l^t\, dj_l$.
  end for
  for $s \in Z_Y^{(l)}$ **do**
    Calculate $C_s^{(l)} = \sum_a^{Z_X^{(l)}} k_{a,s} I_a^{(l)}$.
  end for
  Memoize $\sigma^{(l+1)}(j) = g\left(\sum_b^{Z_Y^{(l)}} j^b C_b^{(l)}\right)$.
end for
The output is given by $\mathcal{F}[\xi] = \sigma^{(L)}$.

Algorithm 2 Error Backpropagation

Input: input γ, desired δ, learning rate α, time t.
for $l \in \{0, \ldots, L\}$ **do**
  Calculate $\Psi^{(l)} = g'\left(\int_{R^{(l-1)}} \sigma^{(l-1)} w^{(l-1)}\, dj_{l-1}\right)$
end for
For every t, compute $\mathfrak{A}_{L,t}$ from from (32).
Update the output coefficient matrix $k_{x,y}^{(L-1)} - I_x^{(L-1)} \int_{R^{(L)}} \left[\mathcal{F}(\gamma) - \delta\right] \Psi^{(L)} j_L^y\, dj_L \to k_{x,y}^{(L-1)}$.
for $l = L-2$ **to** 0 **do**
  If it is null, compute and memoize $\mathfrak{A}_{l+2,t}$ from (32).
  Compute but do not store $\mathfrak{A}_{l+1} \in \mathbb{R}$.
  Compute $\frac{\partial E}{\partial k_{x,y}^{(l)}} = \mathfrak{A}_l$ from from (32).
  Update the weights on layer l: $k_{x,y}^{(l)}(t) \to k_{x,y}^{(l)}$
end for

5. Implementation

With these theoretical guarantees given for $\{\mathcal{G}\}$, the implementation of the feedforward and error backpropagation algorithms in this context is an essential next step. Since the derivations are too long to include here, we defer the reader to other supplemental materials.

Feedforward propagation is straight forward, and relies on memoizing functionals by using the separability of weight polynomials. Essentially, integration need only occur once to yield coefficients on power functions. See Algorithm 1.

For error backpropagation, we chose the most direct analogue for the loss function, in particular since we showed universal approximation using the C^∞ norm, the integral norm will converge.

Definition 13. *For a functional neural network $\mathcal{F}$ and a dataset $\{(\gamma_n(j), \delta_n(j))\}$ we say that the error for a given n is defined by*

$$E = \frac{1}{2} \int_{R^{(L)}} \left(\mathcal{F}(\gamma_n) - \delta_n\right)^2\, dj_L \tag{31}$$

Using this definition we take gradient with respect to the coefficients of the polynomials on each weight surface. Eventially we get a recurrence relation in the same way one might for discrete neaural networks.

$$\begin{aligned}
\mathfrak{B}_{L,t} &= \int_{R^{(L)}} \sum_{b}^{Z_Y^{(L-1)}} k_{t,b}^{(L-1)} j_L^b \left[\mathcal{F}(\gamma) - \delta\right] \Psi^{(L)} \, dj_L. \\
\mathfrak{B}_{s,t} &= \int_{R^{(s)}} \sum_{b}^{Z_Y^{(s-1)}} \sum_{a}^{Z_X^{(s)}} k_{t,b}^{(s-1)} j_s^{a+b} \Psi^{(s)} \mathfrak{B}_{s+1,a} \, dj_s. \\
\mathfrak{B}_{l+1} &= \int_{R^{(l+1)}} \sum_{a}^{Z_X^{(l+1)}} j_{l+1}^{a+y} \Psi^{(l+1)} \mathfrak{B}_{l+2,a} \, dj_{l+1}. \\
\frac{\partial E}{\partial k_{x,y}^{(l)}} = \mathfrak{B}_l &= \int_{R^{(l)}} j_l^x \sigma^{(l)} \mathfrak{B}_{l+1} \, dj_l.
\end{aligned} \tag{32}$$

where Ψ is defined as $g'(T[\sigma] + \beta)$. Using this recurrence relation, we can drastically reduce the time to update each weight by memoizing. That philosophy yields algorithm 2, and therefore we have completed the practical analohgues to these algorithms.

6. Conclusion

In this paper we first extended the standard ANN recurrence relation to infinite dimensional input and output spaces. In this context of this new algorithm, FNN, we proved two new universal approximation theorems. The proposition of functional neural networks lead to new insights into the black box model of traditional neural networks.

Functional networks are a logical generalization of the discrete neural network and therefore all theorems shown for traditional neural networks apply to piecewise functional neural networks. Furthermore the creation of homologous theorems for universal approximation provided a way to find a relationship between the weights of traditional neural networks. This suggests that the discrete weights of a normal artificial neural network can be transformed into continuous surfaces which approximate kernels satisfying the training dataset. We then showed that functional neural networks are also able to approximate bounded linear operators.

The desire to implement $\{\mathcal{F}\}$ in actual learning problems motivated the exploration of a new space of algorithms: $\{G\}$. This new space not only contains standard ANNs and FNNs but also similar extensions such as that proposed in (Roux & Bengio, 2007). We then showed that a subset of $\{G\}$ containg algorithms called continuous classifiers actually reduce the dimensionality of the learning problem when expressed in $\mathfrak{n}_2$ instead of $\mathfrak{n}$ layers.

Finally we proposed computationally feasible error backpropagation and forward propagation algorithms (up to an approximation).

6.1. Future Work

Although we have shown that advantage of using different subset of $\{G\}$ for learning tasks, there is still much work to be done. In this paper we did not explore different classes of weight surfaces, some of which may provide better computational integrability. It was also suggested to us that $\{F\}$ may link kernel learning methods and deep learning. Lastly, it remains to be seen how the general class of $\{G\}$, especially continuous classifiers, can be applied in practice.

References

Burch, Carl. A survey of machine learning. A survey for the Pennsylvania Governor's School for the Sciences, 2001.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–3314, 1989.

Hartig, Donald G. The riesz representation theorem revisited. *The American Mathematical Monthly*, 90(4): pp. 277–280, 1983. ISSN 00029890. URL http://www.jstor.org/stable/2975760.

McCulloch, Warren S and Pitts, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, (4), 1943.

Neal, Radford M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Roux, Nicolas L and Bengio, Yoshua. Continuous neural networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 404–411, 2007.