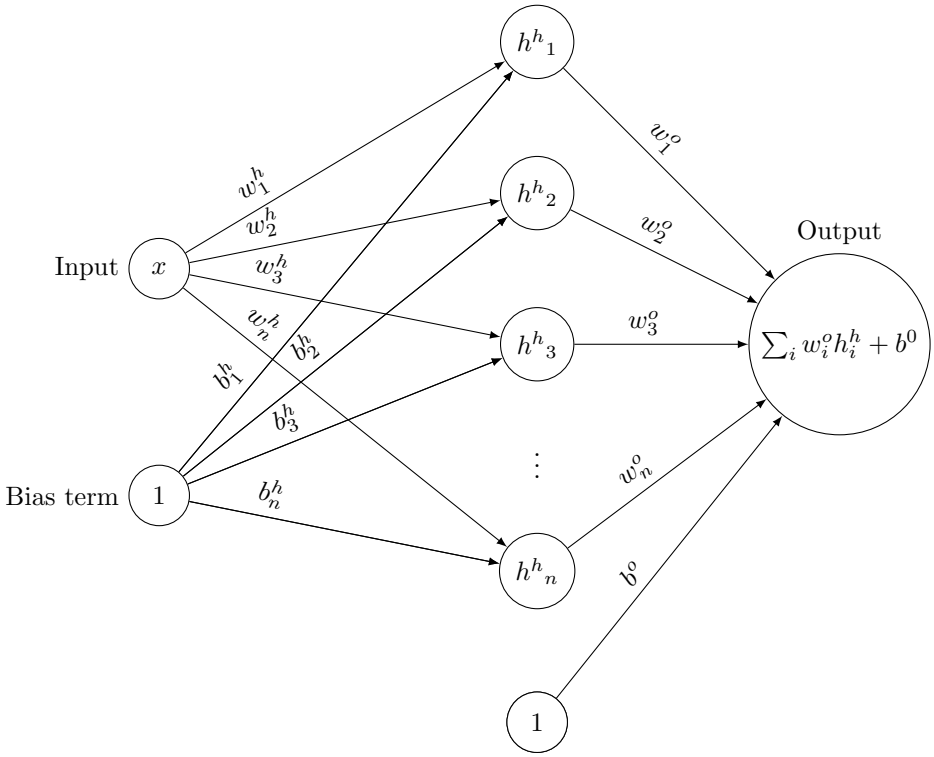
Neural networks are universal approximators

Rafał Skrzypiec



1 Universal approximation theorem

Universal approximation theorem states that a feedforward neural network with one hidden layer and finite but sufficiently large number of neurons can approximate to arbitrary accuracy any functional continuous mapping from one finite-dimensional space to another.

The theorem was proved by Hornik, Stinchcombe and White in 1989 [cytowanie], but before that paper was published, in the same year, Cybenko [cytowanie] had proved universal approximation property for feedforward neural network using sigmoidal activation functions.

Sigmoidal functions is family of functions widely used in Feedforward neural networks, especially for regression purposes.

In this section, I present a proof given by Cybenko in 1989, then I will demonstrate a visual proof of universal approximation theorem using sigmoidal activation functions.

1.1 Approximation by Superpositions of a Sigmoidal Functions

Let I_n denotes the n-dimensional unit cube, $[0,1]^n$. $C(I_n)$ refers to the space of continous functions on I_n. In addition, let $M(I_n)$ denotes space of finite, signed regular Borel measures on n-dimensional unit cube I_n.

Definition 1.1. Measure μ is regural if for every measurable set A, $\mu(A)$ equals the supremum of the measures of closed subsets of A and the infimum of open supersets of A. [Probability measures on metric spaces K.R. Parthasarathy]

Definition 1.2. It is said that $\sigma : \mathbb{R} \to \mathbb{R}$ is sigmoidal if

$$\sigma(x) \to \begin{cases} 1 & \text{as} \quad x \to +\infty \\ 0 & \text{as} \quad x \to -\infty \end{cases}$$

Definition 1.3. It is said that σ is discriminatory if for a measure $\mu \in M(I_n)$

$$\int_{I_n} \sigma\left(w^{\mathsf{T}} x + b_0\right) d\mu(x) = 0 \tag{1}$$

for all $w \in \mathbb{R}$ and $b_0 \in \mathbb{R}$ implies that $\mu = 0$.

Theorem 1.1. *Any bounded, measurable sigmoidal function, σ, is discriminatory. In particular, any continuous sigmoidal function is discriminatory. [cytowanie Cybenko]*

Proof of Cybenko's theorem requires to introduce more useful definitions and theorems. Hahn-Banach theorem is a general theorem which says that any functional can be extended to the whole space while retaining the desired properties.

Theorem 1.2 (Hahn-Banach theorem)**.** *Let X be a real vector space, p a real-valued function defined on X satisfying*

$$p\left(\alpha x+(1-\alpha)y\right)\leq\alpha p(x)+(1-\alpha)p(y)\qquad\forall\alpha\in[0,1],x,y\in X$$

Suppose that λ is a linear functional defined on a subspace $Y\subset X$ which satisfies

$$\lambda(x)\leq p(x)\qquad\forall x\in Y.$$

Then, there is a linear functional Λ, defined on X, satisfying

$$\Lambda(x)\leq p(x)\qquad\forall x\in X,$$

such that

$$\Lambda(x)=\lambda(x)\qquad\forall x\in Y.$$

Reed & Simon (1980), Methods of Modern Mathematical Physics. Functional Analysis

Definition 1.4. The space $\mathcal{L}\left(\mathcal{H},\mathbb{C}\right)$ is called the dual space of Hilbert space $\mathcal{H}$ and is denoted by $\mathcal{H}^*$. The elements of $\mathcal{H}^*$ are called continuous linear functionals.

Reed & Simon (1980), Methods of Modern Mathematical Physics. Functional Analysis

The following important theorem characterizes $\mathcal{H}^*$ and it was formulated by F. Riesz.

Theorem 1.3 (Riesz Representation Theorem)**.** *For each $T\in\mathcal{H}^*$, there is a unique $y_T\in\mathcal{H}$ such that*

$$T(x)=\langle y_T,x\rangle\qquad\forall x\in\mathcal{H}$$

In addition

$$||y_T||_{\mathcal{H}}=||T||_{\mathcal{H}^*}$$

Reed & Simon (1980), Methods of Modern Mathematical Physics. Functional Analysis

Theorem 1.4. *Let σ be any continous discriminatory function. Then finite sums of the form*

$$G\left(x\right)=\sum_{i=1}^{N}w_i^o\sigma\left({w_i^h}^{\mathsf{T}}x+b_i^h\right)\tag{2}$$

are dense in $C(I_n)$. In other words, given any $f\in C(I_n)$ and $\epsilon>0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x)-f(x)|<\epsilon\qquad\forall x\in I_n$$

Proof. Let $S \subset C(I_n)$ be the set of functions of the form $G(x)$ or in other words - set of neural networks. Clearly S is a linear subspace of $C(I_n)$. If S is dense, the closure of S is all of $C(I_n)$.

Assume that closure of S is not all of $C(I_n)$. Then the closure of S, say S', is a closed proper subspace of $C(I_n)$. By the Hahn-Banach Theorem, there exists a bounded linear functional on $C(I_n)$, call it L, with the property that $L \neq 0$ but $L(S') = L(S) = 0$.

By the Riesz Representation Theorem, this bounded linear functional, L, is of the following form

$$L(h) = \int_{I_n} h(x) d\mu(x)$$

for some $\mu \in M(I_n)$, for all $h \in C(I_n)$. In particular, since $\sigma(w^\mathsf{T} x + b)$ is in S' for all w and b, we must have that

$$\int_{I_n} \sigma\left(w^\mathsf{T} x + b\right) d\mu(x) = 0$$

However, we assumed that σ was discriminatory so that this condition implies that $\mu = 0$ contradicting our assumption. Hence, the subspace S must be dense in $C(I_n)$.

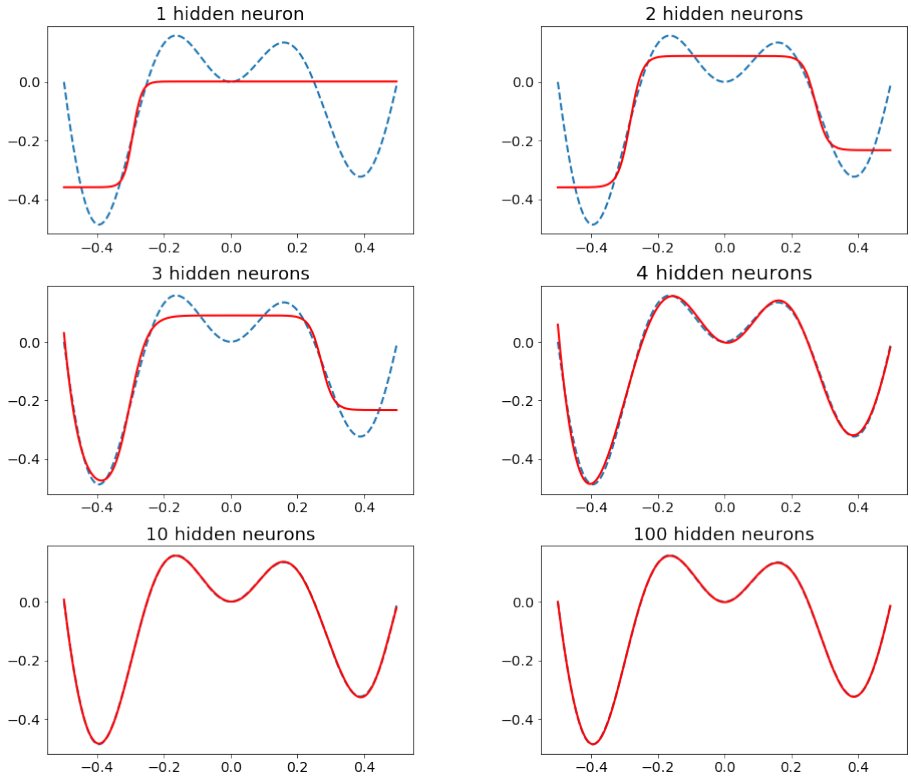
This demonstrates that sums of the form

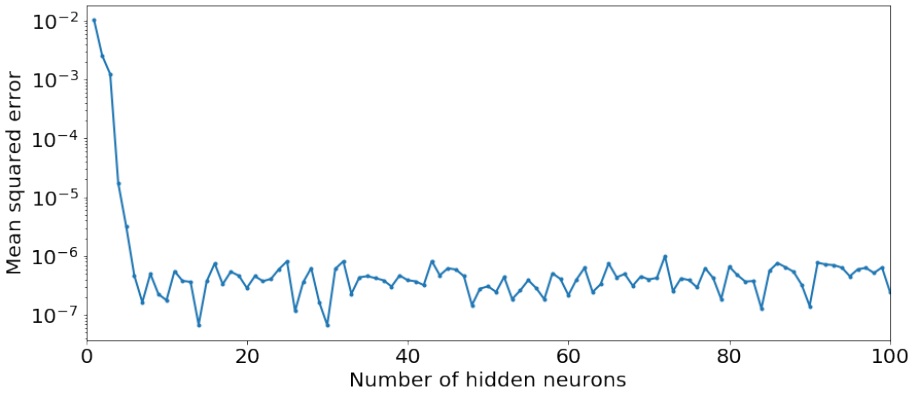
$$G\left(x\right) = \sum_{i=1}^{N} w_i^o \sigma\left(w_i^{h^\mathsf{T}} x + b_i^h\right)$$

are dense in $C(I_n)$ providing that σ is continuous and discriminatory.

□

2 Visual representation





2.1 Data

The training set has m samples of 1 dimension, it is given as a vector: $X \in \mathbb{R}^{1\times m}$ and corresponding results $Y \in \mathbb{R}^{1\times m}$.

2.2 Parameters

The net will have 2 layers: 1) a hidden one, having L neurons, and 2) an output one, having 1 neuron.

The layers are defined through:

1. the parameters of the hidden layer, which maps 1-dimensional input vectors into activations of L neurons: weight matrix $W^h \in \mathbb{R}^{L\times 1}$ and bias vector $b^h \in \mathbb{R}^{L\times 1}$,

2. the parameters of the output layer, which maps L-dimensional vector of activations of the hidden layer to 1 activations of output neurons: weight matrix $W^o \in 1 \times L$ and bias vector $b^o \in \mathbb{R}^{1\times 1}$.

2.3 Signal forward propagation (fprop)

Each hidden neuron computes its total input as a sum of product of its inputs, weight matrix and bias. For an i-th sample, the total input ${a^h}_l^{(i)}$ of an I-th neuron is thus:

$$ {a^h}_l^{(i)} = {W^h}_l x^{(i)} + {b^h}_l \tag{3} $$

The total input of neurons might also be expressed via matrices, using matrix multiplication and broadcasting (which allows to add a column vector to all column vectors of a matrix):

$$ a^h = W^h \cdot x + b^h \tag{4} $$

This can be implemented in Python as $ah = W.dot(x) + b$

Next, we compute activation h^h of hidden neurons with sigmoid $\sigma(a) = \frac{1}{1+e^{-a}}$:

$$ {h^h}_l^{(i)} = \sigma({a^h}_l^{(i)}) \tag{5} $$

Thanks to vectorization in Python + numpy, h^h might be computed with a single expression $hh = numpy.sigmoid(ah)$.

Finally, total input of the output layer can be computed using activations of the hidden layer (with the help of broadcasting) as:

$$ a^o = W^o \cdot h^h + b^o \tag{6} $$

And for I-th sample we have:

$$ \begin{aligned} {a^o}^{(i)} &= {W^o}_l {h^h}_l^{(i)} + b^o \\ &= {W^o}_l \sigma({a^h}_l^{(i)}) + b^o \\ &= {W^o}_l \sigma({W^h}_l x^{(i)} + {b^h}_l) + b^o \end{aligned} \tag{7} $$

We will use mean squared error as the loss function:

$$
\begin{aligned}
J^{(i)}(\Theta) &= \frac{1}{2}\left(y^{(i)} - a^{o(i)}\right)^2 \\
J(\Theta) &= \frac{1}{m}\sum_{i=1}^{m} J^{(i)}(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - a^{o(i)}\right)^2 .
\end{aligned}
\tag{8}
$$

2.4 Error backpropagation (bprop)

Using the chain rule one can derive the gradient of the loss function in respect to neurons' activations and network parameters.

First we compute the gradient with respect to the output layer's total inputs:

$$
\frac{\partial J}{\partial a^{o(i)}} = \frac{1}{m}\left(y^{(i)} - a^{o(i)}\right), \tag{9}
$$

then we compute the gradient with respect to activations of hidden units:

$$
\frac{\partial J}{\partial h^{h}{}_{l}^{(i)}} = \frac{\partial J}{\partial a^{o(i)}}\frac{\partial a^{o(i)}}{\partial h^{h}{}_{l}^{(i)}} = \frac{\partial J}{\partial a^{o(i)}} W^{o}{}_{l}, \tag{10}
$$

then we compute the gradient with respect to the total activations of hidden units:

$$
\frac{\partial J}{\partial a^{h}{}_{l}^{(i)}} = \frac{\partial J}{\partial h^{h}{}_{l}^{(i)}}\frac{\partial h^{h}{}_{l}^{(i)}}{\partial a^{h}{}_{l}^{(i)}} = \frac{\partial J}{\partial h^{h}{}_{l}^{(i)}} h^{h}{}_{l}^{(i)}(1 - h^{h}{}_{l}^{(i)}) \tag{11}
$$

where we have used the relationship

$$
\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))
$$

.

Finally we can use the gradients with respect to the total inputs to compute the gradients with respect to network parameters, eg. for the input layer:

$$
\frac{\partial J}{\partial W^{o}{}_{l}} = \sum_{i}\frac{\partial J}{\partial a^{o(i)}}\frac{\partial a^{o(i)}}{\partial W^{o}{}_{l}} = \sum_{i}\frac{\partial J}{\partial a^{o(i)}} h^{h}{}_{l}^{(i)}, \tag{12}
$$

$$
\frac{\partial J}{\partial b^{o}} = \sum_{i}\frac{\partial J}{\partial a^{o(i)}}\frac{\partial a^{o(i)}}{\partial b^{o}} = \sum_{i}\frac{\partial J}{\partial a^{o(i)}}. \tag{13}
$$