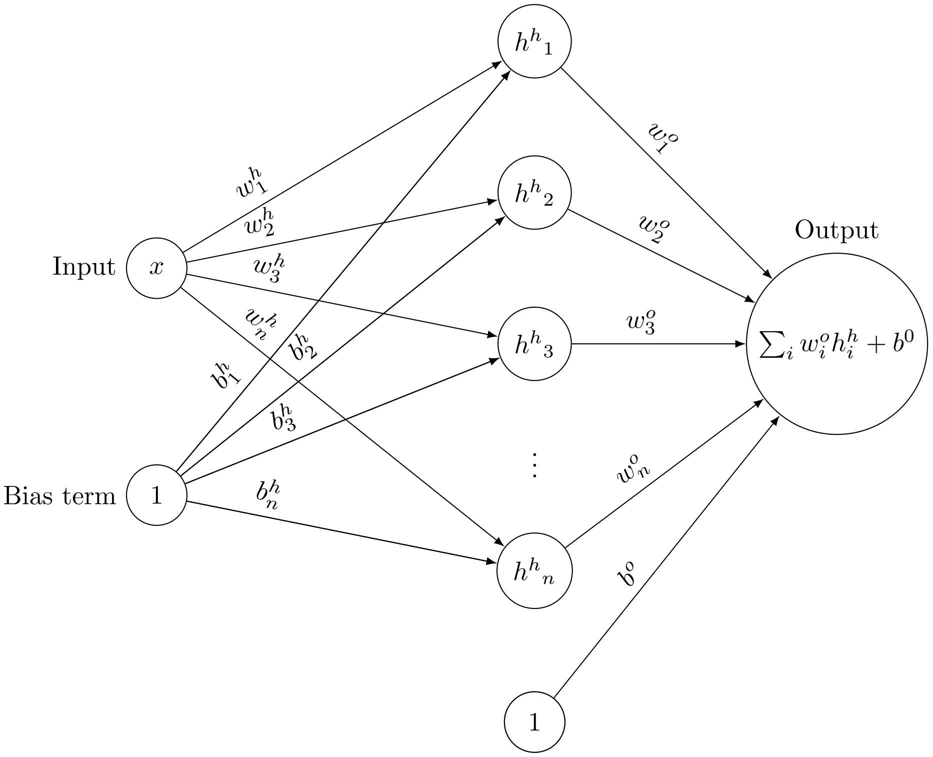
Neural network is universal approximator, why sigmoids?

Rafal̈ Skrzypiec



zamieni y na w i zamienic θ na b

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. Zobaczymy czy si przyda $f : I_n \to C(I_n)$, $||f|| = \sup |f(x)| : x \in I_n$.

$||f||$ is used to denote the supremum norm of an $f \in C(I_n)$.

Definition 1.3. 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 1.4. It is said that σ is discriminatory if for a measure $\mu \in M(I_n)$

$$\int_{I_n} \sigma\left(y^T x + \theta\right) d\mu(x) = 0$$

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

————————————————-====TU SKONCZYLEM

Hahn-Banach theorem shows how to extend linear functionals from subspaces to whole spaces. Moreover, we can do it in a way that respects the boundedness properties of the given functional. The most general formulation of the theorem requires a preparation

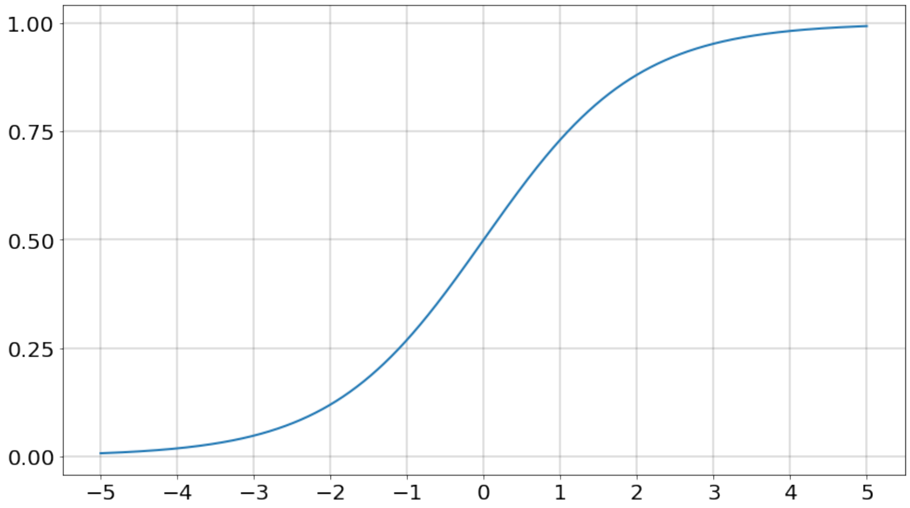


Figure 1: Example of sigmoidal function - sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$

Definition 1.5. A sublinear functional is a function $f : V \to \mathbb{R}$ on a vector space V which satisfies subadditivity (1) and positive homogenity conditions (2)

$$f(x+y) \leq f(x) + f(y) \qquad \forall x, y \in V \tag{1}$$

$$f(\alpha x) = \alpha f(x) \qquad \forall \alpha \geq 0, x \in V \tag{2}$$

Theorem 1.1 (Hahn-Banach theorem for real vector spaces)**.** *If $p : V \to \mathbb{R}$ is a sublinear function, and $\psi : U \to \mathbb{R}$ is a linear functional on a linear subspace $U \subset V$, and satisfying $\psi(x) \leq p(x)$ $\forall x \in U$. Then there exists a linear extension $\Psi : V \to \mathbb{R}$ of ψ to the whole space V, such that*

- $\Psi(x) = \psi(x)$ $\forall x \in U$
- $\Psi(x) \leq p(x)$ $\forall x \in V$

Rudin 1991, Th 3.2

Theorem 1.2 (Riesz representation theorem)**.** *Let H be a Hilber space over $\mathbb{R}$, and T a bounded linear functional on H. If T is a bounded linear functional on a Hilbert space H then there exist some $g \in H$ such that for every $f \in H$ we have (http://www.math.jhu.edu/ lindblad/632/riesz.pdf)*

$$T(f) = \langle f, g \rangle \qquad \forall f \in H$$

Any bounded linear functional T on the space of compactly supported continuous functions on X is the same as integration against a measure μ. (http://mathworld.wolfram.com/RieszRepres

$$Tf = \int f d\mu$$

Theorem 1.3. *Let σ be any continous discriminatory function. Then finite sums of the form*

$$G\left(x\right)=\sum_{j=1}^{N}\alpha_{j}\sigma\left(w_{j}^{T}x+\theta_{j}\right)$$

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 whic

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

Proof. Let $S\subset C(I_n)$ be the set of functions of the form $G(x)$. Clearly S is a linear subspace of $C(I_n)$. We claim that 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 R, is a closed proper subspace of $C(I_n)$. By the Hahn-Banach theorem, there is a bounded linear functional on $C(I_n)$, call it L, with the property that $L\neq 0$ but $L(R)=L(S)=0$.

By the Riesz Representation Theorem, this bounded linear functional, L, is of the 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 $\sigma(y^Tx+\theta)$ is in R for all y and θ, we must have that

$$\int_{I_n}\sigma\left(y^Tx+\theta\right)d\mu(x)=0$$

for all y and θ.

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

This demonstrates that sums of the form

$$G(x)\sum_{j=1}^{N}\alpha_{j}\sigma\left(y_{j}^{T}x+\theta_{j}\right)$$

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

□

2 Visual proof of

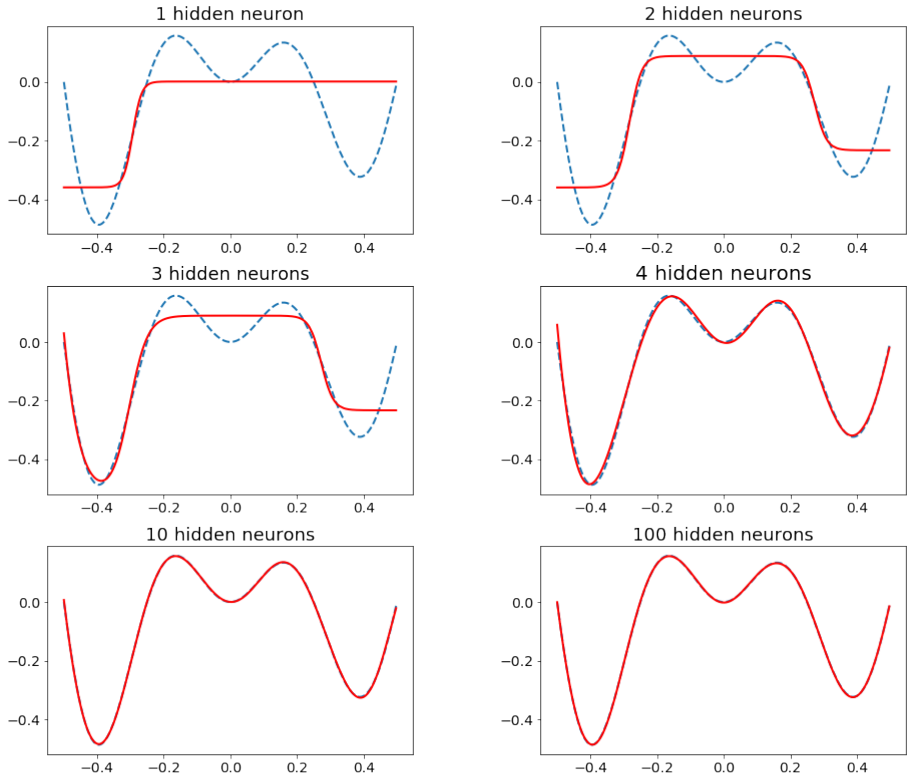


Figure 2: Visual proof, wydrukowa MSE i narysowa dla 10 neuronow i 100, INNA FUNKCJA!!

3 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 $Y \in \mathbb{R}^{1\times m}$.

4 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 $b^o \in \mathbb{R}^{1\times 1}$.

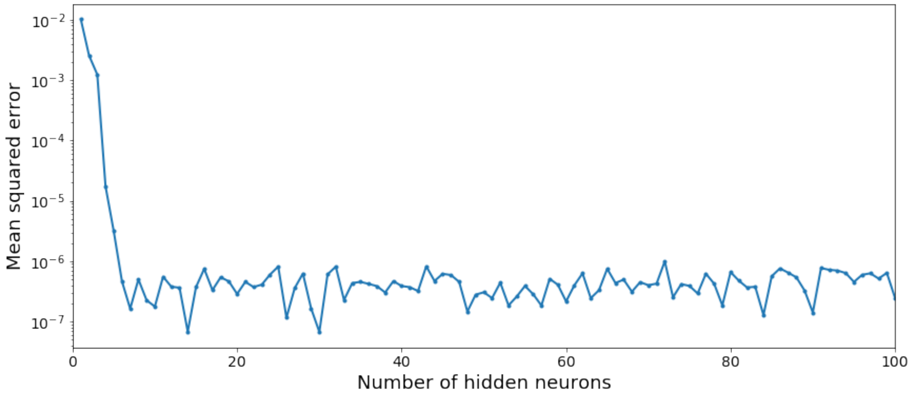


Figure 3:

Number of hidden neurons	Mean squared error
1	0.00942516615858
2	0.00277472585580
3	0.00135999931016
4	5.33546419190e-05
10	1.70243920789e-06
100	1.11243069317e-06

5 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:

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

We will use mean squared error as the loss function:

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

6 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:

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

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

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

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:

$$\frac{\partial J}{\partial {W^o}_l} = \sum_i \frac{\partial J}{\partial a^{o(i)}} \frac{\partial a^{o(i)}}{\partial {W^o}_l} = \sum_i \frac{\partial J}{\partial a^{o(i)}} h^{h^{(i)}_l}, \tag{12}$$

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

7 Probabilistic interpretation of sigmoid

In section 4.2 of Pattern Recognition and Machine Learning (Springer 2006), Bishop shows that the logit arises naturally as the form of the posterior probability distribution in a Bayesian treatment of two-class classification. He then goes on to show that the same holds for discretely distributed features, as well as a subset of the family of exponential distributions. For multi-class classification the logit generalizes to the normalized exponential or softmax function. Following this, the value of the logit or softmax can therefore actually be interpreted as a probability in a variety of settings, but not as the frequentist probability of an event, but as the Bayesian probability of an underlying cause (class) given the data.