Neural networks and the universal approximation theorem

Neural networks

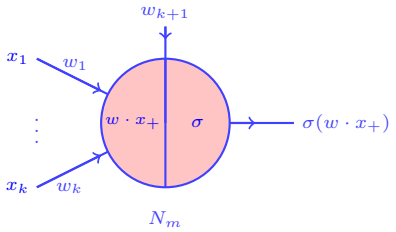
Def. Given
1. a function $\sigma : \mathbf{R} \to \mathbf{R}$
2. a vector $w \in \mathbf{R}^{k+1}$

we consider the **neuron** $N_{\sigma,w}$ to be the function defined by

$$\begin{array}{rccc} N_{\sigma,w} : & \mathbf{R}^k & \to & \mathbf{R} \\ & x & \mapsto & \sigma(w \cdot x_+) \end{array}$$

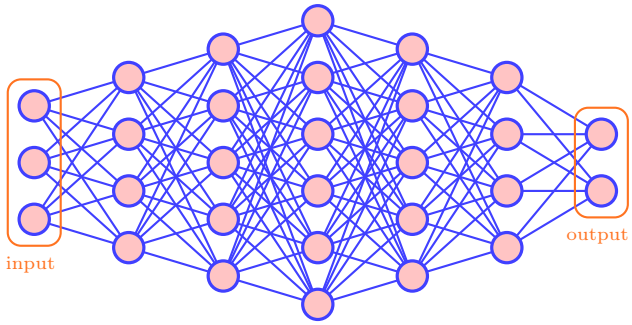
where if $x = (x_1, \dots, x_k)$ then x_+ denotes the vectors $x_+ = (x_1, \dots, x_k, 1)$.

The function σ is said to be the **activation function** of the neuron, and w is said to be its **weight vector**.



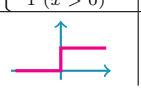
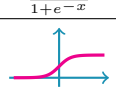
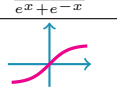
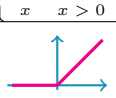
Def.
A **layer** of (σ, k)-neurons of height l is a function $\mathbf{R}^k \to \mathbf{R}^l$ given by a tuple $L = (N_{\sigma,w_1}, \dots, N_{\sigma,w_l})$ of neurons of length l (where all the w_i's are in $\mathbf{R}^{k+1}$).

Def. A **layered neural network** of shape $(k_1, k_2, \dots, k_n)$ is the composition (from right to left) of a tuple $(L_0, \dots, L_n)$ of layers of (σ_i, k_i)-neurons of height l_i (where $l_i = k_{i+1}$ for all $1 \leq i \leq n-1$).



Note
- The left-most layer L_0 is called the **input layer**. Its neurons are $(\mathrm{id}, 1)$-neurons with weight vector $(1, 0)$.
- The right-most layer L_n is called the **output layer**. Its neurons are often have an activation function that differs from the other layers.
- The other layers are called **hidden layers**. They often share the same activation function ($\sigma_1 = \cdots = \sigma_{n-1}$).

Most used activation functions :

Identity	Heavyside	Sigmoid	Hyperbolic tan	ReLu
x	$\begin{cases} 0 & (x \leq 0) \\ 1 & (x > 0) \end{cases}$	$\frac{1}{1+e^{-x}}$	$\frac{e^x - e^{-x}}{e^x + e^{-x}}$	$\begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$
				

Rem. There are more sophisticated neural networks :
- **Convolutional neural networks** are used in visual imagery. Their first hidden layer applies a change of basis in the space of pictures (= list of pixels) in order to recognize simple patterns, then the second layer searches for simple patterns of simple patterns, and the third one searches for patterns of patters of patterns and so on. These first layers are followed by a standard neural network as above.
- **Recurrent neural networks** allow for loops between the layers (so some connections go from layer 7 to layer 3 for example). They are able to learn to recognize context-sensitive languages.
- **Differentiable neural computers** are memory augmented neural networks. They can learn basic algorithmic tasks like sorting, copying, error correcting (during communications) and so on. They are proved to be Turing complete.

Perceptrons

Def. A **perceptron** is a neural network in which the activation functions in the hidden layers are Heavyside functions and in which all the inputs are either 1 or 0. Thus they implement functions from $f : \mathbf{F}_2^n \to \mathbf{F}_2^m$.

Theorem (McCulloch & Pitts, 1943) any logical function $f : \mathbf{F}_2^n \to \mathbf{F}_2^m$ is representable by a perceptron.

Proof
Clearly it is enough to prove the theorem for functions $f : \mathbf{F}_2^n \to \mathbf{F}_2$ (since we can superpose neural networks, one for each partial function).

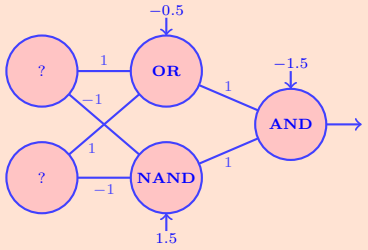
For a moment, let us assume that our neurons can be logical gates (**AND** , **OR** , **NOT**).

Then for each vector $v \in \mathbf{F}_2^n$, it is easy to build, for each $v \in \mathbf{F}_2^n$, and using only using only **AND** and **NOT** gates a neural network with only one output neuron that returns 1 iff the input is v. Let us superpose all the selectors of those v's in $f^{-1}(1)$.

We obtain a network with a large number of neurons at the output. If the input is chosen in in $f^{-1}(0)$, all of these output neurons will return 0. Otherwise exactly one of the output neurons will return 1.

So using **OR** gates to sum up all of these output neurons we obtain a network whose values coincide with the values of f.

So we are left with proving that neurons can implement logical gates. But this is easy. Here for example is the **XOR** gate :



Training neural networks

Rosenblatt's algorithm

In 1958, Rosenblatt showed that a simple algorithm allows to train a formal neuron (i.e. a neuron with a Heavyside activation function). The principle is simple : assume given a subset of inputs $X \subset \{0,1\}^n \subset \mathbf{R}^n$, a function $h : X \to \{0,1\}$, and assume $X_0 = h^{-1}(0)$ and $X_1 = h^{-1}(1)$ are separated in $\mathbf{R}^n$ by an affine hyperplane.

The algorithm :
Input : $(\mathbf{d}_0, f_0)$, $(\mathbf{d}_1, f_1)$, ... , $(\mathbf{d}_N, f_N)$ with $f_i = 0$ or 1 so that $\mathbf{d}_i \in X_{f_i}$.

```
Initialization :   choose w randomly
                   set finished=False

Main loop :        while not finished :
                       repeat Epoch

Epoch :            set finished=True
                   for i = 0, ..., N :
                       if (w.d_i > 0) == f_i :
                           pass
                       else :
                           set finished=False
                           w += (-1)^(1+f_i) d_i
```

This simple algorithm allows, for example, to train a neuron to classify pictures of triangles and circles.

Unfortunately, it soon appeared that
- affine separation is too much to ask in general
- no such algorithm could train a deeper network like the one for the **XOR** gate above.

Backpropagation of errors

In order to overcome the above problems, in the 80's, people began to use sigmoïdal activation functions. For a sample $(x_1, \dots, x_N)$ of elements in $\mathbf{R}^n$, a function $f : \mathbf{R}^n \to \mathbf{R}$, and a sigmoïdal neural network NN, we define, for a choice $\mathbf{w}$ of weights :

$$\varepsilon(\mathbf{w}) = \sum_{i=1}^{N} [f(x_i) - \mathrm{NN}_{\mathbf{w}}(x_i)]^2$$

Then ε is a differentiable function and we can use **gradient descent** to find some of its local minima. This gradient descent is made efficient by a clever application of the chain rule for derivatives known as the backpropagation of errors algorithm.

The universal approximation theorem
Copied (and lightly adaptated) from Leshko, Lin, Pinkus, Schocken

Fact 1
Distributions that have a vanishing higher derivative are polynomials :

$$\left[\ u(\varphi^{(k+1)}) = 0 \quad \forall \varphi \in \mathcal{C}_c^\infty \ \right] \quad \Leftrightarrow \quad [\ u \in \mathbf{R}_k[X]\ ]$$

$\Leftarrow$ is obvious.

Let us prove $\Rightarrow$ by induction.

The case $k = -1$ is obvious : if $u(\varphi) = 0$ for all test function φ then $u = \mathbf{0}$ ($\mathbf{0}$ denotes the null distribution).

The case $k = 0$ will be needed in the induction step. So let us prove it : consider u such that $u' = 0$. The distribution $\mathbf{1}$ is defined by $\varphi \mapsto \int_{\mathbf{R}} \varphi$. Choose a test function ψ with $\int_{\mathbf{R}} \psi = 1$. Then for any φ,

$$\xi = \varphi - \mathbf{1}(\varphi)\psi$$

is the derivative of a test function (since its integral vanishes). So our hypothesis implies that $u(\xi) = 0$, so that $u(\varphi) = \mathbf{1}(\varphi)u(\psi)$. Hence $u = u(\psi)\mathbf{1}$ is a constant.

Now assume it is known that if $u^{(k)} = 0$ then $u \in \mathbf{R}_{k-1}[X]$ and pick a distribution u for which $u^{(k+1)} = 0$. Then by the induction hypothesis, we know that $u' \in \mathbf{R}_{k-1}[X]$. Let $P(X)$ be this polynomial, and let $Q(X)$ be its primitive that vanishes at 0. Consider

$$w = u - Q$$

Then our hypothesis implies that w has a vanishing derivative, so is constant by the $k = 0$ case.

Fact 2
Let σ be L^1_{loc}. If $\sigma * \varphi$ is a polynomial for any test function φ then $\sup_\varphi(d^\circ \sigma * \varphi) < +\infty$

This is an application of Baire's Category Theorem : for any $[a, b]$ we endow $\mathcal{C}_0^\infty([a, b]$ with the metric

$$d(\varphi, \psi) = \sum_n \frac{\|\varphi - \psi\|_n}{2^n(1 + \|\varphi - \psi\|_n)} \quad \text{where} \quad \|f\|_n = \sum_{j=0}^{n} \max_{[a,b]} |f^{(j)}|$$

This distance makes it a Fréchet space, in which we can apply Baire's theorem. Set $V_k = \{\varphi, d^\circ(\sigma * \varphi) \leq k\}$. This is an exhaustive union of closed nested subspaces, so one of them must be the whole space $\mathcal{C}_0^\infty([a, b]$.

To obtain the result on the whole $\mathbf{R}$ we note that the maximal degree of these polynomials we found on $\mathcal{C}_0^\infty([a, b]$ is translation invariant. Then using a partition of the unit, we see that the maximal degree on a union of two intervals of the same size equals the maximal degree on one of the intervals. Iterating, we get the result.

The conjunction of these two facts gives :

Theorem : Let σ be L^1_{loc}. If $\sigma * \varphi$ is a polynomial for any test function φ then σ itself is a polynomial.

$$\left[\ \forall \varphi \in \mathcal{C}_c^\infty, \quad \exists k, \quad \sigma * \varphi^{(k+1)} = 0 \ \right] \quad \Leftrightarrow \quad [\ \exists n, \quad \sigma \in \mathbf{R}_n[X]\ ]$$

Def.
Let M be the space of functions that are continuous (a.e.) and bounded (a. e.) on every compact.

For a function $\sigma \in M$ we let $S(\sigma)$ denote the vector space spanned by the functions $x \mapsto \sigma(ax + b)$ for a and b in $\mathbf{R}$.

A function f is said to belong to $\bar{S}(\sigma)$ if for any compact $K \subset \mathbf{R}$, there exists a sequence of elements of $S(\sigma)$ that converges to f on K (for the L^∞ norm on K).

Fact 3
For a given $\sigma \in M$ and for any test function φ, $\sigma * \varphi$ belongs to $\bar{S}(\sigma)$.

Let $[a, b]$ be an interval. The idea is to approximate φ by step functions st_n (functions that are constant on $[a + i\frac{b-a}{n}, a + (i + 1)\frac{b-a}{n}])$ for $i = 0, \ldots, n - 1)$. For such a step function, $\sigma * st_n$ belongs to $S(\sigma)$. When n tends to infinity, this sequence of elements of $S(\sigma)$ tends to $\sigma * \varphi$ thus showing that $\sigma * \varphi$ belongs to $\bar{S}(\sigma)$.

Fact 4 If $\sigma \in \mathcal{C}^\infty$ is not a polynomial then any function $f \in \mathcal{C}(\mathbf{R})$ belong to $\bar{S}(\sigma)$

The idea is to prove that under the hypothesis, any polynomial belongs to $\bar{S}(\sigma)$. Then by the Weierstrass approximation theorem, any continuous function belongs to $\bar{S}(\sigma)$.

This follows from the fact that $x \mapsto \frac{\sigma((a+h)x+b)-\sigma(ax+b)}{h}$ belongs to $S(\sigma)$, so that, taking the limit as h tends to 0, we get that $x\sigma'(ax+b)$ belongs to $\bar{S}(\sigma)$ for any a and b. Choosing $a = 0$, we get that $x\sigma'(b)$ belongs to $\bar{S}(\sigma)$ for any b. Now σ not being a polynomial, there exists a b for which $\sigma(b) \neq 0$ so x belongs to $\bar{S}(\sigma)$.

The same argument, using k-th derivatives instead of first derivatives, shows that x^k belongs to $\bar{S}(\sigma)$ for any k.

The conjunction of the previous theorem and of these facts shows :

Theorem : For any function $\sigma : \mathbf{R} \to \mathbf{R}$ that is continuous (a.e.) and bounded on any compact (a.e.), and is not a polynomial, any function $f \in C(R)$ belongs to $\bar{S}(\sigma)$.

Fact 5 The span of all functions $x \mapsto f(w \cdot x_+)$ for $f \in \mathcal{C}(\mathbf{R})$ and $w \in \mathbf{R}^{k+1}$ is dense in $\mathcal{C}(\mathbf{R}^k)$.

Using Weierstrass again, it suffices to show that all the monomials $x_1^{d_1} \cdots x_k^{d_k}$ belong to this span. This is a simple exercise in linear algebra (just counting dimensions).

Fact 6 If σ is a polynomial then $S(\sigma)$ is finite dimensional and cannot be dense in $\mathcal{C}(\mathbf{R})$.

Using Weierstrass again, it suffices to show that all the monomials $x_1^{d_1} \cdots x_k^{d_k}$ belong to this span. This is a simple exercise in linear algebra (just counting dimensions).

Putting everything together, we get :

Theorem : Let $\sigma : \mathbf{R} \to \mathbf{R}$ be continuous (a.e.) and bounded on any compact (a.e.). Then the following are equivalent :

any continuous function on $\mathbf{R}^k$ can be arbitrarily well approximated on any compact by an element of $S(\sigma)$

σ is not a polynomial.

Finally, let μ be a measure on $\mathbf{R}^k$, that is finite and has compact support K. Since $\mathcal{C}(K)$ is dense in $L^p(\mu)$ we get :

Corollary : Let $\sigma : \mathbf{R} \to \mathbf{R}$ be continuous (a.e.) and bounded on any compact (a.e.). Then the following are equivalent :

any function $f \in L^p(\mu)$ can be arbitrarily well approximated (for the L^p metric) by an element of $S(\sigma)$

σ is not a polynomial.

Conclusion

If σ is not a polynomial, any function in $\mathcal{C}(\mathbf{R}^k)$ (resp $L^p(\mu)$ for a finite measure with compact support μ on $\mathbf{R}^k$) can be arbitrarily well approximated by a neural network with

- one input layer with identity activation functions,
- one hidden layer with (a potentially very large number of neurons having) σ as activation function,
- one neuron with identity activation function as its output layer.