Fourier Neural Networks

Adrian Silvescu
Artificial Intelligence Research Group
Department of Computer Science
Iowa State University, Ames, IA 50010
Email:silvescu@cs.iastate.edu

Abstract

A new kind of neuron model that has a Fourier-like IN/OUT function is introduced. The model is discussed in a general theoretical framework and some completeness theorems are presented. Current experimental results show that the new model outperforms by a large margin both in representational power and convergence speed the classical mathematical model of neuron based on weighted sum of inputs filtered by a nonlinear function. The new model is also appealing from a neurophysiological point of view because it produces a more realistic representation by considering the inputs as oscillations.

1 Introduction

The first mathematical model of a neuron was proposed by McCulloch & Pitts[1943]. The underlying idea that this model tries to capture is that the response function of a neuron is a weighted sum of its inputs filtered through a nonlinear function: $y = h(\sum w_i x_i + \theta)$.

Much progress has been done in the field of neural networks since that time but this idea still remained a very fundamental one. Although the model of the computational unit(neuron) *per se* is simple, neural networks are powerful computers, higher levels of complexity being achieved by connecting many neurons together.

In this paper we try to propose more general and powerful models for the neuron as a computational unit. There are may motivations for this investigation.

One of them is the fact that although the power of computers increased quite a lot since 1943 we are still not able to simulate and train but toy-size neural networks. So although from a theoretical point of view creating complexity out of very basic components is desirable, from a practical point of view more powerful models of the computational units(neurons) are more appealing because they can help reduce the size of the networks by some orders of magnitude and are also more suitable to coarse grained paralelization. More complex

TRADING SOFTWARE

FOR SALE & EXCHANGE

www.trading-software-collection.com

***Subscribe** for FREE download more stuff.*

Mirrors:

www.forex-warez.com
www.traders-software.com

Contacts

andreybbrv@gmail.com
andreybbrv@hotmail.com
andreybbrv@yandex.ru
Skype: andreybbrv
ICQ: 70966433

and powerful computational imply also a more compact representation of the information stored in the network, making it an improvement from an Occam razor point of view.

Another motivation towards more general and elaborated models neurons comes from the discoveries in neurobiology that show more that more complex phenomena take place at the neuron level.

Although apparently different from the early model of McCulloch&Pitts our model is still based on the same kind of idea (although in a more general way) "of computing the output of the neuron as weighted sum of the activations produced by the inputs".

We will first introduce a general framework and discuss some of the issues that appear. Then a particular model, the Fourier Neural Networks is introduced and closely examinated. Next some specific theoretical results are presented followed by experimental results. Finally the conclusions and further development are discussed. *Note:* The Fourier Neural Networks were introduced in Silvescu[1997].

2 A general framework

In this section we will introduce a general model for the computation function of the neuron. We begin with a continuous model that will be further discretised for computational reasons.

2.1 The model

Let $\mathbf{x} = (x_1, ..., x_n) \in \mathbf{R}^n$, $\mathbf{y} = (y_1, ..., y_m) \in \mathbf{R}^m$ *and* $D \subseteq \Re^m$ *then the output function of the neuron on input* $\mathbf{x}$ *will be given by*

$$f(\mathbf{x}) = \int_D c(\mathbf{x})\phi(\mathbf{x}, \mathbf{y})d\mathbf{y} \tag{1}$$

What does this formula mean? - The output function f is a sum of some characteristics of the input $\mathbf{x}$ given by $\phi(\mathbf{x}, \mathbf{y})$ wheighted by the coefficients $c(\mathbf{y})$. So in a certain sense the old principle of weighted sum of the inputs still applies only that in a more general way, inputs being replaced by some characteristics of of them $\phi(\mathbf{x}, \mathbf{y})$.

If we would like to have our outputs range in $[0, 1]$, as in many traditional neural networks we can transform the previous formula into:

$$f(\mathbf{x}) = h(\int_{\mathbf{D}} c(\mathbf{x})\phi(\mathbf{x}, \mathbf{y})d\mathbf{y}) \tag{2}$$

Where h is a nonlinear function such as the sigmoid:

$$h(x) = \frac{1}{1 + e^{-tx}}$$

Without loss of generality but for the sake of simplicity we will concentrate our discussion on the equation (1), the results being easily extendable also for the equation (2).

Equation (1) is a particular case of what it is called in functional analysis an integral operator. An integral operator U following Kantorovitch[1992] is defined as follows:

Definition *Let S, T two spaces with measures an let* $\mathbf{X}$ *and* $\mathbf{Y}$ *two functions spaces on S and T respectively. Then an operator* $U : \mathbf{X} \rightarrow \mathbf{Y}$ *is called an integral operator if there exists a measurable function K(s,t) such that, for every* $x \in \mathbf{X}$*, the image* $y = U(x)$ *is the function:*

$$y(s) = \int_T K(s,t)x(t)dt$$

The function K(s,t) is called the kernel of U.

Depending on the particular choice of the kernel K we can get different function subspaces of $\mathbf{Y}$ as an image of U, $U(\mathbf{X})$. Since our goal is get powerful computational units the we will seek kernels that produce "extensive" images $U(\mathbf{X})$ of operator U.

2.2 Some solution for equation (1)

Mathematical Analysis provides a few solutions for the equation (1) that we will examine next:

2.2.1 Fourier integral formula

For every $f \in L_2(\mathbf{R})$ we have:

$$f(t) = \int_{\mathbf{R}} c(\omega)e^{i\omega t}d\omega$$

Where

$$c(\omega) = \frac{1}{2\pi}\int_{\mathbf{R}} f(\tau)e^{-i\omega\tau}d\tau$$

And $\sqrt{2\pi}c(\omega)$ is the Fourier transform of the function f on ω. This formula has been extracted from Walker[1988].

2.2.2 The windowed Fourier integral formula

For every $f \in L_2(\mathbf{R})$ we have:

$$f(t) = \int_{\mathbf{R}}\int_{\mathbf{R}} c(\omega,\theta)g(t-\theta)e^{i\omega t}d\omega d\theta$$

Where

$$c(\omega,\theta) = \frac{1}{2\pi}\int\limits_{\mathbf{R}} f(\tau)g(\tau-\theta)e^{-i\omega\tau}d\tau$$

And $\sqrt{2\pi}c(\omega,\theta)$ is the Fourier transform of the function f, with the window g centered in θ, on ω. This formula has been extracted from Gasquet[1990]. An example of window function is given in the figure 1.

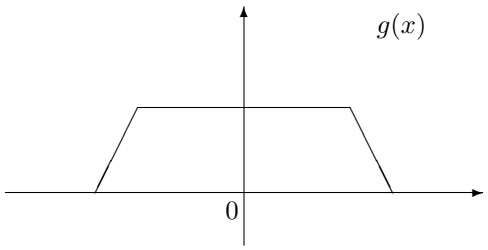

Figure 1: **Example of window function**

2.2.3 The wavelets integral formula

For every $f \in L_2(\mathbf{R})$ we have:

$$f(t) = \int\limits_{\mathbf{R}}\int\limits_{\mathbf{R}} c(a,b)\psi(a,b,t)dadb$$

Where

$$c(a,b) = \frac{1}{a^2}C_\psi^{-1}\int\limits_{\mathbf{R}} f(\tau)\psi(a,b,\tau)d\tau$$

$$\psi(a,b,t) = |a|^{-1/2}\psi\left(\frac{t-b}{a}\right)$$

$$C_\psi = 2\pi\int\limits_{\mathbf{R}} |\xi|^{-1}|\hat{\psi}(\xi)|^2 d\xi$$

And $\hat{\psi}(\xi)$ is the Fourier transform of the function ψ on ξ. with ψ satisfying $C_\psi < \infty$.

$a^2C_\psi c(a,b)$ is called the Wavelet transform of the function f on $\psi(a,b)$ centered in b and having the scale a.

An example of the ψ function is the second derivative of the gaussian (also called mexican hat):

$$\psi(t) = \frac{2}{\sqrt{3}}(1 - x^2)e^{x^2/2}.$$

This formula has been extracted from Daubechies[1992].

2.3 The multidimensional extension

Although the previous formulae are for functions defined on one-dimensional spaces, they can be naturally extended for many dimensions. We will only give the multidimensional extension for the Fourier integral formula that is of particular interest for us.

$$f(x_1, ...x_n) = \int\limits_{\mathbf{R}^n} c(\omega_1, ..., \omega_n)e^{i(\omega_1 x_1 + ...\omega_n x_n)}d\omega_1...d\omega_n$$

Where

$$c(\omega_1, ..., \omega_n) = \frac{1}{(2\pi)^n} \int\limits_{\mathbf{R}^n} f(t_1, ...t_n)e^{-i(t_1 x_1 + ...t_n x_n)}dt_1...dt_n$$

2.4 What can we do on a computer?

We can discretize equation (1) by approximating the integral using the rectangular or trapezoidal rule or Riemann sums in the following way:

$$f^d(\mathbf{x}) = \sum_{y_1...y_n} c'(y_1, ..., y_n)\phi(y_1, ..., y_n, \mathbf{x}) \tag{3}$$

Where

$$c'(y_1, ..., y_n) = \delta(y_1, ..., y_n)c(y_1, ..., y_n) \tag{4}$$

Where $\delta(y_1, ..., y_n)$ is the measure of the interval centered in $(y_1, ..., y_n)$. *Note:* $c'(y_1, ..., y_n)$ represents the factor that is not dependent on $\mathbf{x}$ in the Riemann sums.

Observation: Because of the way integral was defined it follows that for every function f that can be written of the form given by equation (1) there exists a sequence of functions $(f_n^d)_{n\in\mathbf{N}}$ that converges pointwise to f.

2.5 Recapitulation

In this section we introduced a new model for the neuron as a computational unit given by the equation (1). Then we discretized the equation (1) and we obtained a discrete model given by equation (3) that can approximate arbitrarily well our continuous model (pointwise convergence). This model is a very general one and can be used for every solution of equation (1).

In the next section, we will focus on a particular model (the Fourier integral formula) for which we will obtain further properties in the following sections.

3 Fourier Neural Networks

3.1 The neuron model

In the case of the Fourier integral formula the discretized model will be given by the following equation:

$$f^d(\mathbf{x}) = \sum_{y_1 \dots y_n} c'(y_1, ..., y_n) e^{i(y_1 x_1 + ... + y_n x_n)} \tag{5}$$

We will modify this formula in two ways in order to make it more suitable for computer implementation. First we will use the *cosinus* representation for $e^{i(y_1 x_1 + ... + y_n x_n)}$, and we will also filter the output through the sigmoid function in order to obtain outputs in the interval $[0, 1]$. (Note: This last step is optional)

So the final form of the output function for our neuron is:

$$f(x_1, ...x_n) = sigmoid(\sum_i c_i \prod_{j=1}^{n} cos(\omega_{ij} x_j + \varphi_{ij}))$$

3.2 Neurophysiological justification

From a neurophysiological point of view, the Fourier neural networks are closer to reality because they model the signals exchanged between neurons as oscillations making our model to agree better with discoveries made in neurobiology. The same remark can be also made about wavelets and windowed Fourier integral formula which model the signals as spikes of oscillation. Our neuron model implements also a type of neural architecture discovered in the brain called $\Sigma\Pi$ units.

3.3 Comparison with the Discrete Fourier Transform(TFD)

For every function f we can associate a Fourier series and for functions satisfying certain properties we can write

$$f(x_1, ..., x_n) = \sum_{m_1, ..., m_n} c(m_1 ... m_n) e^{i(m_1 x_1 + ... + m_n x_n)}$$

Where $m_1, ..., m_n \in \mathbf{N}$ and

$$c(m_1, ..., m_n) = \frac{1}{(2\pi)^n} \int_{\mathbf{R}^n} f(\mathbf{y}) e^{-i<\mathbf{m},\mathbf{y}>} d\mathbf{y}$$

Where $\mathbf{m} = (m_1, ..., m_n)$ and $\mathbf{y} = (y_1, ..., y_n)$.

The main difference between computing the Discrete Fourier Transform(DFT) and adapting the neuron using the gradient descent method is statistical versus exact information. Let us consider for example a function that is composed out of two oscillations of phase 0 and frequencies $2\pi 9/20$ and $2\pi 11/20$ and let us

make a Discrete Fourier Transform using $2\pi/10$ as the main frequency. Then the plot of the Fourier coefficients against the frequency multiplicator n will be a function that has all the coefficients nonzero. The adaptive method should find the two frequencies and set the coefficients to the corresponding amplitudes. In this way adaptive method offers an exact frequency information versus a statistical information given by the DFT.

4 Theoretical results

In this section we will enunciate two theorem from Walker[1988] concerning the convergence Multidimensional Fourier Series and we will comment on the implications of these theorems regarding the power of representation of our model of neuron.

Theorem. *(MSE convergence)* The complex exponential system $\{e^{i<\mathbf{m},\mathbf{x}>}\}$ (where $\mathbf{m} = (m_1, ..., m_n)$ and $\mathbf{x} = (x_1, ..., x_n)$) is complete in the space of n-dimensional continuos functions on $[-\pi, \pi]^n$. That is for every continuous function f on $[-\pi, \pi]^n$ we have

$$\lim_{m_1,...,m_n \to \infty} \int_{[-\pi,\pi]^n} |f(\mathbf{x}) - S_{\mathbf{m}}(\mathbf{x})|^2 d\mathbf{x} = 0$$

Where $S_{\mathbf{m}}$ is the partial Fourier series corresponding to the limits $\mathbf{m}$.

Theorem. *(Uniform convergence)* If $f : \mathbf{X} \to \mathbf{Y}$ is continuous with period 2π in every subset of components then the Fourier series for f converges uniformly to f.

Since our neurons are capable of representing any partial Fourier series it follows that the previous theorems hold for also for or neuron models giving us a characterization of their power of representation.

So in conclusion (with a minor changes of the previous theorems) we have that our neurons are capable of approximating arbitrarily well every continuous function defined on $[0, 1]^n$.

5 The implementation and experimental results

The training method for our neural networks was gradient descent with momentum plus a factor given by the convergence accelerator that is discussed in next.

5.1 The convergence accelerator

The code for the convergence accelerator is the following:

```
if (abs(d)<acsm) and (d<>0)
  then
    acs:=(1/m)*factor*sgn(d)*((acsm*acsm)/d)
```

```
else
  acs:=0;
```

The basic idea of the convergence accelerator is the following: If the last adjustment of the weights (gradient + momentum) is to very small then we are on a plateau and we are going to introduce an extra adjustment the is inverse proportional with the value of the normal adjustment(the flatter the plateau is the faster we will go).

This convergence accelerator seems to improve by a few orders of magnitude the convergence speed and it is also useful for getting out of local minima.

5.2 Experimental results

The main thing we looked after in our experiments was to try to get an estimate of the power of representation and the convergence speed.

The Fourier Neural Networks have been tested on all the 512 boolean functions that can be defined on matrices 3x3 and on all the 256 boolean functions with 3 variables (using only two terms in the sum of equation (5)) and managed to converge on average more than 100 times faster than the classical neural networks using 8 neurons in the hidden layer.

We also tested the Fourier Neural Networks on all the 65536 booolean functions that can be defined over matrices of 4x4. The functions have been learned on average after 1.921 random initializations. More exact figures are presented in the following table: (the fist column representing the number of random initializations, the second column the percentage of functions learned using that number of initialization, the third column is the cumulative percentage of the functions learned and the fourth column is the cumulative number of functions learned out of the total of 65536).

```
 1 : 64.30%   64.30%   42142
 2 : 17.33%   81.63%   53498
 3 :  4.87%   86.50%   56690
 4 :  6.05%   92.55%   60656
 5 :  2.88%   95.43%   62543
10 :  0.26%   99.13%   64968
20 :  0.02%   99.90%   65472
34 :  0.00%   99.99%   65527
65 :  0.00%  100.00%   65536
```

6 Conclusions and further work

We proposed very general framework that is suitable for studying models that belong to the paradigm of "weighted sum of input characteristics" that generalizes the model proposed by McCulloch&Pitts[1943] . We also explored in more detail a particular model that belongs to this paradigm: the Fourier Neural Networks, that proved to be appealing from neurobiological, theoretical and practical point of view.

Further work we believe could investigate the behavior of other kernel functions (such as wavelets and windowed Fourier but not only). Another interesting direction would be toproduce extensive reports of experimental results on real world applications, that are the ultimate test for every learning scheme.

References

[1992] Daubechies I., 1992, *Ten Lectures on Wavelets.*

[1990] Gasquet C., Witomski P., 1990, *Analise de Fourier et applications.*

[1988] Walker J., 1988, *Fourier Analysis.*

[1998] Harris J. W., Stocker H., 1998, *Handbook of Mathematics and Computational Science.*

[1992] Kantorovitch L.V., Akilov G.P., 1992, *Functional Analysis.*

[1997] Silvescu A., 1997, *A new kind of neural networks,* Master disertation thesis.

[1943] McCulloch W. S., Pitts W., (1943), *A new kind of neural networks,* Master disertation thesis.