Fourier Analysis and Filtering of a Single Hidden Layer Perceptron *

Robert J. Marks II, Payman Arabshahi
Department of Electrical Engineering, University of Washington FT–10
Seattle, WA 98195 USA

Abstract— We show that the Fourier transform of the linear output of a single hidden layer perceptron consists of a multitude of line masses passing through the origin. Each line corresponds to one of the hidden neurons and its slope is determined by that neuron's weight vector. We also show that convolving the output of the network with a function can be achieved simply by modifying the shape of the sigmoidal nonlinearities in the hidden layer.

1 Preliminaries

Consider a layered perceptron with a single hidden layer of H hidden neurons. The output, $\theta(\vec{x})$, of the perceptron is linear and the input is a vector $\vec{x}$ of dimension N. An example for the case $N = 2$, $H = 3$ is shown in Fig. 1(a). We have, for the general case:

$$\theta(\vec{x}) = \sum_{h=1}^{H} a_h \sigma(\vec{w}_h^T \vec{x} + \phi_h), \tag{1}$$

where $\vec{w}_h^T = [w_{h1}\, w_{h2}\, \ldots\, w_{hN}]$, $\vec{x}^T = [x_1\, x_2\, \ldots\, x_N]$, T denotes transposition, and generally $\sigma(\cdot)$ is the sigmoid function,

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2}$$

In the analysis to follow, however, $\sigma(z)$ can be any function (e.g. Gaussian).

Carroll & Dickinson (1989) showed that such networks can implement an arbitrarily good approximation to any $\mathcal{L}_2$ function over $[-1, 1]^n$. The functional form of the output of such a network was shown to be a finitely parameterized, approximate form of the back-projection operator, a component of the inverse Radon transform (Deans, 1983).

Ito (1993) extended this result to cases where the domain of approximation is either a compact subset of the Euclidean space, or its entirety. Approximations of m-times continuously differentiable functions in several variables, and their derivatives were considered.

Our charter here is to examine the Fourier transform of the output of such a network. We assume that the neural network is trained and that all weights are fixed.

2 Theory

Define the N dimensional Fourier transform (see for example (Marks, 1991)) of $\theta(\vec{x})$ as:

*(invited paper) International Conference on Artificial Neural Networks (IEEE/ENNS), Sorrento, Italy, May 1994.

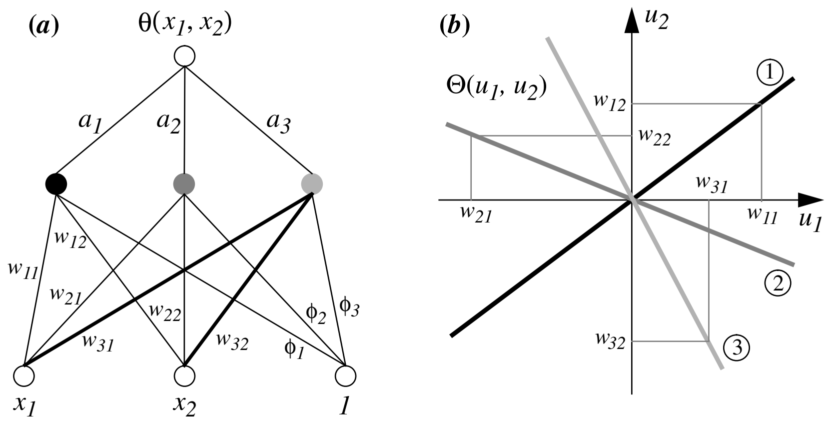

Figure 1: (a) Example for $N = 2$ inputs and $H = 3$ hidden neurons. The bold connections from the input to hidden layer are components of the interconnect vector $\vec{w}_3$. (b) Plot of $\Theta(\vec{u})$ for the network in (a). There are 3 lines corresponding to a delta function "slice" for each hidden neuron. The lines' slopes are determined by each hidden neuron's weight vector.

$$\Theta(\vec{u}) = \int_{\vec{x}} \theta(\vec{x}) e^{-j2\pi\vec{u}^T\vec{x}} d\vec{x} \tag{3}$$

where

$$\int_{\vec{x}} = \int_{x_1=-\infty}^{\infty} \int_{x_2=-\infty}^{\infty} \cdots \int_{x_N=-\infty}^{\infty},$$

$d\vec{x} = dx_1 dx_2 \cdots dx_N$, and $\vec{u}^T = [u_1\, u_2\, \ldots\, u_N]$. The notation $\theta(\vec{x}) \leftrightarrow \Theta(\vec{u})$ is sometimes used to denote Fourier transform pairs. Thus:

$$\Theta(\vec{u}) = \sum_{h=1}^{H} a_h \int_{\vec{x}} \sigma(\vec{w}_h^T\vec{x} + \phi_h) e^{-j2\pi\vec{u}^T\vec{x}} d\vec{x}. \tag{4}$$

We arrive at the following expression for $\Theta(\vec{u})$ (see the Appendix):

$$\Theta(\vec{u}) = \sum_{h=1}^{H} \frac{a_h}{|w_{hN}|} \Sigma(\frac{u_N}{w_{hN}}) e^{j2\pi\phi_h u_N / w_{hN}} \delta(\vec{u}_- - \frac{u_N}{w_{hN}} \vec{w}_{h-}), \tag{5}$$

where $\Sigma(\cdot)$ is the Fourier transform of $\sigma(x)$ [1], $\vec{w}_{h-}^T = [w_{h1}\, w_{h2}\, \ldots\, w_{h(N-1)}]$, $\vec{x}_-^T = [x_1\, x_2\, \ldots\, x_{N-1}]$, $\vec{u}_-^T = [u_1\, u_2\, \ldots\, u_{N-1}]$ and $\delta(\vec{x}) = \delta(x_1)\delta(x_2)\ldots\delta(x_N)$ where $\delta(\xi)$ is the impulse or Dirac delta function. $\Theta(\vec{u})$ is seen to be the sum of H scaled delta functions in N space. Each hidden neuron corresponds to a different line mass in N space. This parallels the observations of Carroll & Dickinson (1989). The back-projection of an object's projection, when transformed, results in the line masses shown in Fig. (1).

To illustrate, consider the case of a 2 dimensional input vector with 3 hidden neurons as shown in Fig. 1. There are three line masses corresponding to the 3 hidden neurons. Each line's slope is determined by the corresponding hidden neuron weight vector (see Eq. (32)).

[1] For the nonlinearity in Eq. (2), $\Sigma(u) = \frac{1}{2}\delta(u) - j\pi \mathrm{cosech}(2\pi^2 u)$.

If the output of the neural network is now convolved with a function $\gamma(\vec{x})$, the only effect will be a change in the weighting of its line masses. Their orientation, and hence all $\{\vec{w}_h\}$, will not change. We now examine this case in more detail.

3 Convolution

Consider the case where the output of the neural network is convolved with a function $\gamma(\vec{x})$. The methodology we present here can be used as an alternative to training neural networks with jitter to improve generalization (Reed, 1992).

We will show that if [2]

$$\theta(\vec{x}) = \sum_{h=1}^{H} a_h \sigma(\vec{w}_h^T \vec{x}), \tag{6}$$

and

$$\varphi(\vec{x}) = \theta(\vec{x}) * \gamma(\vec{x}), \tag{7}$$

then

$$\varphi(\vec{x}) = \sum_{h=1}^{H} a_h \sigma_\gamma(\vec{w}_h^T \vec{x}) \tag{8}$$

where

$$\sigma_\gamma(y) \leftrightarrow \Sigma_\gamma(u) \qquad \gamma(\vec{x}) \leftrightarrow \Gamma(\vec{\rho})$$

are Fourier transform pairs, and

$$\Sigma_\gamma(u) = \Sigma(u)\Gamma(\vec{w}_h u). \tag{9}$$

Convolving the output of the neural network with a function $\gamma(\vec{x})$ is therefore equivalent to convolving the sigmoidal nonlinearities with the same function.

Proof:

$$\begin{aligned}
\varphi(\vec{x}) &= \int_{\vec{\xi}} \theta(\vec{x} - \vec{\xi})\gamma(\vec{\xi})d\vec{\xi} && (10)\\
&= \sum_h a_h \int_{\vec{\xi}} \sigma(\vec{w}_h^T \vec{x} - \vec{w}_h^T \vec{\xi})\gamma(\vec{\xi})d\vec{\xi} && (11)\\
&= \sum_h a_h \sigma_\gamma(\vec{w}_h^T \vec{x}). && (12)
\end{aligned}$$

where:

$$\sigma_\gamma(y) = \int_{\vec{\xi}} \sigma(y - \vec{w}_h^T \vec{\xi})\gamma(\vec{\xi})d\vec{\xi}. \tag{13}$$

Thus:

$$\begin{aligned}
\Sigma_\gamma(u) &= \int_y \left[\int_{\vec{\xi}} \sigma(y - \vec{w}_h^T \vec{\xi})\gamma(\vec{\xi})d\vec{\xi}\right] e^{-j2\pi u y}dy && (14)\\
&= \Sigma(u) \int_{\vec{\xi}} \gamma(\vec{\xi})e^{-j2\pi u \vec{w}_h^T \vec{\xi}}d\vec{\xi} && (15)\\
&= \Sigma(u)\Gamma(\vec{w}_h u). && (16)
\end{aligned}$$

[2] Inclusion of the bias term ϕ_h in Eq. (6) simply produces a multiplicative phase term $\exp(-j2\pi\phi_h u)$ on the right hand side of Eq. (9). It is omitted to simplify notation.

3.1 Examples

<u>HILBERT TRANSFORM</u>

The Hilbert transform in one dimension (Bracewell, 1986) is obtained by convolving a function with the kernel $\gamma(x) = -(\pi x)^{-1}$. Extending this to N dimensions, we have:

$$\gamma(\vec{x}) = \prod_{n=1}^{N} \frac{-1}{\pi x_n}. \tag{17}$$

Taking an N dimensional Fourier transform, we obtain

$$\Gamma(\vec{\rho}) = \prod_{n=1}^{N} j\,\mathrm{sgn}(\rho_n), \tag{18}$$

where sgn$(\cdot)$, the signum function, is equal to 1 for positive arguments and -1 for negative arguments. Thus, from Eq. (9), we have:

$$\Sigma_\gamma(u) = \Sigma(u)[j\,\mathrm{sgn}(u)]^N \prod_{n=1}^{N} \mathrm{sgn}(w_{hn}), \tag{19}$$

where w_{hn} is the nth component of $\vec{w_h}$. For $N = 1$ this reduces to:

$$\begin{aligned} \Sigma_\gamma(u) &= \mathrm{sgn}(w_{h1})\,j\,\mathrm{sgn}(u)\Sigma(u) \\ &= \mathrm{sgn}(w_{h1})\,\mathcal{H}\{\sigma(y)\} \end{aligned} \tag{20}$$

where $\mathcal{H}\{\sigma(y)\}$ represents the Hilbert transform of $\sigma(y)$.

Furthermore if $w_{h1} > 0 \quad \forall\, h$, the above relationship reduces to $\Sigma_\gamma(u) = \mathcal{H}\{\sigma(y)\}$. Thus taking the Hilbert transform of the output of the neural network corresponds to replacing each nonlinearity in the hidden layer neurons with its Hilbert transform.

<u>DIFFERENTIATION</u>

Differentiation, commonly used in neural networks for sensitivity analysis (Minai & Williams, 1993; Oh et. al., 1991, Priddy et. al., 1993), can be viewed as a convolution with a unit doublet. In one dimension, taking the derivative of a function is equivalent to multiplying the function's Fourier transform with $(j2\pi\rho)$, where ρ is the frequency variable. Extending this to M dimensions, we have the M dimensional derivative kernel

$$\Gamma(\vec{\rho}) = \prod_{m=1}^{M} (j2\pi\rho_m), \tag{21}$$

where we are performing a derivative with respect to the first M input variables $(M \leq N)$. Extension to higher order derivatives is straighforward.

Thus from Eq. (9), we have:

$$\Sigma_\gamma(u) = (j2\pi u)^M (\prod_{m=1}^{M} w_{hm})\Sigma(u), \tag{22}$$

and

$$\sigma_\gamma(y) = (\prod_{m=1}^{M} w_{hm})\frac{\partial^M \sigma(y)}{\partial y^M}. \tag{23}$$

In one dimension this reduces to:

$$\sigma_\gamma^h(y) = w_{hP}\frac{\partial \sigma(y)}{\partial y}. \qquad h = 1, 2, \ldots H \tag{24}$$

where we have replaced $\sigma_\gamma(y)$ with $\sigma_\gamma^h(y)$ to identify the sigmoid function corresponding to the hth hidden neuron. The index P ($1 \leq P \leq N$, where N is the input dimension) refers to the input variable x_P with respect to which the first order derivative is being taken. It is seen that taking the derivative of the output of the neural network is equivalent to the following two operations: (a) Replacement of the sigmoidal functions in the hidden layers by their derivatives, and (b) Replacement of the set of weights $\{a_h\}$ by $\{a_h w_{hP}\}$ for $h = 1, 2, \ldots H$.

References

Bracewell, R. (1986) *The Fourier transform and its applications.* McGraw-Hill: New York.

Carroll, S.M., and Dickinson B.W. (1989) "Construction of neural nets using the Radon transform," *Proc. International Joint Conference on Neural Networks,* Washington DC, USA, vol. 1, pp. 608–611.

Deans, S.R. (1983) *The Radon transform and some of its applications.* John Wiley & Sons: New York.

Ito, Y. (1993) "Radon transform and differentiable approximation by neural networks," *Proc. International Joint Conference on Neural Networks,* Nagoya, Japan, vol. 3, pp. 2288–2291.

Marks, R.J. II (1991) *Introduction to Shannon sampling and interpolation theory.* Springer-Verlag: New York.

Minai A. A. and Williams R. (1993) "On the derivatives of the sigmoid," *Neural Networks,* vol. 6, no. 6, pp. 845–853.

Oh, S, Marks, R. J. II, and El-Sharkawi, M. A. (1991) "Query based learning in a multilayer perceptron in the presence of data jitter," *Proc. First International Forum on Applications of Neural Networks to Power Systems,* Seattle, WA USA (IEEE), pp. 72–75.

Priddy K. L., Rogers S. K., Ruck D. W., Tarr G. L., and Kabrisky M. (1993) "Bayesian selection of important features for feedforward neural networks," *Neurocomputing,* vol. 5, no. 2–3, pp. 91–103.

Reed, R, Marks, R.J. II, and Oh, S. (1992) "An equivalence between sigmoidal gain scaling and training with noisy (jittered) input data," *Proc. RNNS/IEEE Symposium on Neuroinformatics and Neurocomputing,* Rostov-on-Don, Russia, pp. 120–127.

Appendix

We have from Eq. (4):

$$\int_{\vec{x}} \sigma(\vec{w}_h^T \vec{x} + \phi_h) e^{-j2\pi\vec{u}^T\vec{x}} d\vec{x}$$
$$= \int_{\vec{x}_-}\int_{x_N} \sigma(\vec{w}_{h-}^T \vec{x}_- + w_{hN}x_N + \phi_h) e^{-j2\pi(\vec{u}_-^T\vec{x}_- + u_N x_N)} dx_N d\vec{x}_- \tag{25}$$

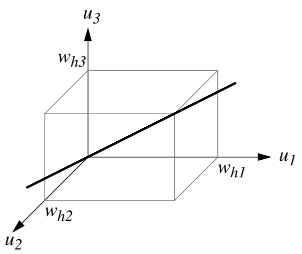

Figure 2: Plot of Eq. (32) for $N = 3$

Letting $z = \vec{w}_{h-}^T \vec{x}_- + w_{hN} x_N + \phi_h$ we obtain:

$$\int_{\vec{x}} \sigma(\vec{w}_h^T \vec{x} + \phi_h) e^{-j2\pi\vec{u}^T\vec{x}} d\vec{x} \tag{26}$$

$$= \int_{\vec{x}_-} \int_z \sigma(z) e^{-j2\pi(\vec{u}_-^T \vec{x}_- + (z - \vec{w}_{h-}^T \vec{x}_- - \phi_h) u_N / w_{hN})} \frac{dz}{w_{hN}} dx_-$$

$$= \frac{1}{|w_{hN}|} \int_{\vec{x}_-} \left[\int_z \sigma(z) e^{-j2\pi z u_N / w_{hN}} dz \right]$$

$$\times e^{-j2\pi(\vec{u}_-^T - \vec{w}_{h-}^T u_N / w_{hN})\vec{x}_-} d\vec{x}_- e^{j2\pi\phi_h u_N / w_{hN}} \tag{27}$$

Defining:

$$\Sigma(u) = \int_{-\infty}^{\infty} \sigma(z) e^{-j2\pi u z} dz, \qquad \delta(\vec{u}_-) = \delta(u_1)\delta(u_2)\ldots\delta(u_{N-1}),$$

We have:

$$\int_{\vec{x}} \sigma(\vec{w}_h^T \vec{x} + \phi_h) e^{-j2\pi\vec{u}^T\vec{x}} d\vec{x}$$

$$= \frac{1}{|w_{hN}|} e^{j2\pi\phi_h u_N / w_{hN}} \Sigma(\frac{u_N}{w_{hN}}) \int_{\vec{x}_-} e^{-j2\pi(\vec{u}_-^T - \vec{w}_{h-}^T u_N / w_{hN})\vec{x}_-} d\vec{x}_- \tag{28}$$

$$= \frac{1}{|w_{hN}|} e^{j2\pi\phi_h u_N / w_{hN}} \Sigma(\frac{u_N}{w_{hN}}) \, \delta(\vec{u}_- - \vec{w}_{h-} u_N / w_{hN}). \tag{29}$$

Equation (29) is nonzero only when

$$\vec{u}_- = \vec{w}_{h-} u_N / w_{hN}, \tag{30}$$

or, an element at a time:

$$u_n = w_{hn} u_N / w_{hN}; \qquad 1 \leq n \leq N - 1 \tag{31}$$

The above equation can be written as:

$$\frac{u_1}{w_{h1}} = \frac{u_2}{w_{h2}} = \ldots = \frac{u_{N-1}}{w_{h(N-1)}} = \frac{u_N}{w_{hN}}. \tag{32}$$

In N-space, this defines a line. For example, if $N = 3$, we have the situation in Fig. 2. Note that the line goes through the origin and the point $\vec{u} = \vec{w}_h$.

Substituting Eq. (29) into Eq. (4) results in Eq. (5).