Author(s): Palmer-Brown, Dominic; Kang, Miao
Article title: Deep Learning in an Adaptive Function Neural Network
Year of publication: 2006
Citation: Palmer-Brown, D. and Kang, M. (2006) 'Deep Learning in an Adaptive Function Neural Network' Proceedings of the AC&T, pp.78-83.
Link to published version:
http://www.uel.ac.uk/act/proceedings/documents/ACT06Proceeding.pdf

DEEP LEARNING IN AN ADAPTIVE FUNCTION NEURAL NETWORK

Dominic Palmer-Brown, Miao Kang*
Innovative Informatics Research Group;
(d.palmer-brown@uel.ac.uk,).
**Leeds Metropolitan University*

Abstract: Artificial neural network learning is typically accomplished via adaptation between neurons. This paper describes adaptation that is simultaneously between and within neurons. The conventional neurocomputing wisdom is that by adapting the pattern of connections between neurons the network can learn to respond differentially to classes of incoming patterns. The success of this approach in an age of massively increasing computing power that has made high speed neurocomputing feasible on the desktop and more recently in the palm of the hand, has resulted in little attention being paid to the implications of adaptation within the individual neurons. The computational assumption has tended to be that the internal neural mechanism is fixed. However, there are good computational and biological reasons for examining the internal neural mechanisms of learning. Recent neuroscience suggests that neuromodulators play a role in learning by modifying the neuron's activation function [Scheler] and with an adaptive function approach it is possible to learn linearly inseparable problems fast, even without hidden nodes. The ADaptive FUction Neural Network (ADFUNN) presented in this paper is based on a linear piecewise neuron activation function that is modified by a novel gradient descent supervised learning algorithm [Palmer-Brown;Kang]. It has been applied to the Iris dataset, and a natural language phrase recognition problem, exhibiting impressive generalisation classification ability with no hidden neurons.

1. Motivation

True to its name the artificial neuron derives from a joint biological-computational perspective. Summing weighted inputs is biologically plausible, and adapting a weight is a reasonable model for synaptic modification. In contrast, the common assumption of a fixed output activation function is for computational rather than biological reasons. A fixed analytical function facilitates mathematical analysis to a greater degree than an empirical one. Nonetheless, there are some computational benefits to modifiable activation functions, and they may be biologically plausible as well. Recent neuroscience suggests that neuromodulators play a role in learning by modifying the neuron's activation function [Scheler]. From a computational point of view, it is surprising if real neurons are essentially fixed entities with no adaptive aspect, except at their synapses, since such a restriction leads to non-linear responses requiring many neurons.

Multi-Layer Perceptrons (MLPs) can be very effective, but if the activation function is not optimal, neither is the number of hiddens which in turn depends on the function.

Adapting a slope-related parameter of the activation function may help, but not if the analytic shape of the function is unsuited to the problem domain. In contrast, with an adaptive function approach it should be possible to learn linearly inseparable problems fast, even without hidden nodes.

In this paper, we improve the general

learning rule of ADFUNN by using proximal-proportional function adaptation. The simplest application of ADFUNN is the two input exclusive-OR. A single classic artificial neuron is incapable of solving it, yet with an adaptive activation function, the solution is readily learnable with one neuron [Palmer-Brown].

Another case is the Iris dataset [Fisher] which consists of 150 four dimensional data. This linearly inseparable problem can be solved by a 4 input, 3 output ADFUNN [Palmer-Brown] network without a hidden node. The generalisation reaches 100% with 80% of the testing patterns, within 100 epochs.

We also perform natural language phrase recognition on a set of phrases from the Lancaster Parsed Corpus (LPC) [Garside]. Generalisation rises to 100% with 150 training patterns (out of a total of 254).

The learned functions resemble noisy versions of characteristic shapes, so we use another ADFUNN to recognize (and therefore select) the best matched analytical function to the empirical learned one in each neuron. Before doing this, all noisy data should be removed from the functions. In this paper, we compare two different algorithms for smoothing the learned function curves, the simple moving average and least-squares polynomial smoothing.

2. Learning problems tackled with ADFUNN

We provide a means of solving linearly inseparable problems using a simple adaptive function neural network (ADFUNN), based on a single layer of linear piecewise function neurons, as shown in figure 1.

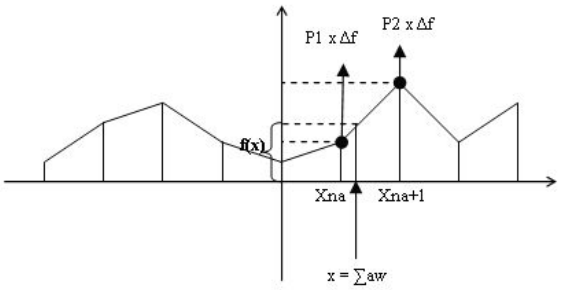

Fig. 1. Adapting the linear piecewise neuron activation function in ADFUNN

We calculate ∑aw, and find the two neighboring f-points that bound ∑aw. However, in our previous work, two f-points are adapted together if ∑aw is approximately equidistant from them. In this paper, the two proximal f-points will be adapted separately, on a proximal-proportional basis. The proximal-proportional value P1 is (Xna+1 - x)/(Xna+1 – Xna) and value P2 is (x - Xna)/(Xna+1 - Xna). Thus, the change to each point will be in proportion to its proximity to x. We obtain the output error and adapt the two proximal f-points separately, using a function modifying version of the delta rule, as outlined in 2.1 to calculate Δf.

2.1. The General Learning Rule

The weights and activation functions are adapted in parallel, using the following algorithm:

A = input node activation, E = output node error.

WL, FL: learning rates for weights and functions.

Step1: calculate output error, E, for input, A.

Step2: adapt weights to each output neuron:

$\Delta w = WL \; Fslope \; A \; E$

$w' = w + \Delta w$

weights normalisation

Step3: adapt function for each output neuron:

$$\Delta f\ (\sum aw) = FL\ E$$

$$f'_1 = f_1 + \Delta f\ P1,\ f'_2 = f_2 + \Delta f\ P2$$

Step4: $f\ (\sum aw) = f'\ (\sum aw)$; $w = w'$.
Step5: randomly select a pattern to train

Step6: repeat step 1 to step 5 until the output error tends to a steady state.

2.2. Iris dataset

The Iris dataset [Fisher] consists of 150 four dimensional data. Four measurements: sepal length, sepal width, petal length, and petal width, were made by Fisher on 50 different plants from each of three species of Iris (Iris Setosa, Iris Versicolor and Iris Virginica). One class is linearly separable from the other two, but the other two are not linearly separable from each other.

We apply a 221 input, 4 output ADFUNN without hidden neurons to solve this problem. On more than 100 simulations (it learns the task on no more than 30 epochs in each run), the correct classification always reaches 100% with 120 training patterns and 30 testing patterns. An example function output is as follows, in this case for Iris Setosa class:

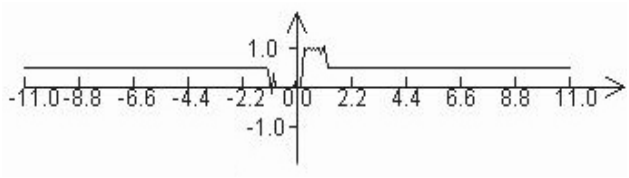

Fig. 2. Iris Setosa function output in ADFUNN (f-points are initialized to a constant value (0.5) making it easy to identify the active range over which adaptation has occurred.)

We can interpret the data by considering weights and functions together. To identify Iris Setosa, both petal width and petal length must be taken into account as they both have equally strong weights. Thus the NN is easy to interpret (transparent): for a given range of petal width plus petal length, as indicated by the interval between [0.2, 1.2] in the above figure, the flower is Setosa.

2.3. Phrase Recognition

We generated 254 input patterns using the pre-tagged corpus from the Lancaster Parsed Corpus (LPC). A total of 49 bits are used to encode all possible input symbols. The terminal symbol groups are: punctuation (Pu), conjunctions (Co), nouns (NP), verbs (VP) and prepositions (PP). The non-terminal symbol groups are sentences (S), finite clauses (F) non-finite clauses (T), major phrase types (V) and minor phrase types (M). There are 4 look-back symbols, 10 phrasal symbols and 1 look-ahead symbol, which makes a total of 15 inputs symbols. Thus, the total number of inputs is 49 bits x 15 symbols = 735. According to LPC there are 41 constituent tags altogether. Thus we have a 735 inputs and 41 outputs network to deal with the phrase recognition, using 254 input patterns. The following is the learned function output for sentence:

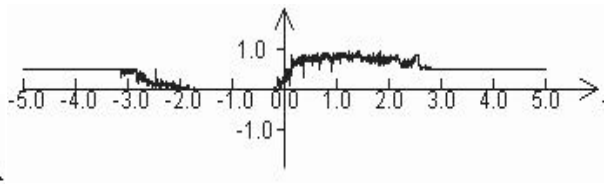

Fig. 3. Sentence function output for phrase recognition using ADFUNN (f-points are initialized to a constant value (0.5) making it easy to identify the active range over which adaptation has occurred.)

The generalisation reaches 100% with 150 training patterns and 104 testing patterns. We compare the performance of ADFUNN, with an MLP connectionist parser phrase

recognizer trained on a much larger (superset) dataset, and with an MLP simple back-propagation network trained on the same dataset. The most obvious advantage of ADFUNN is the lack of hidden neurons, whereas 50 hidden layer nodes are required in the other two networks. Additionally, ADFUNN achieves 100% correct classification compared to 98.76% and 89.01% for the best MLP networks.

3. Function Curve Smoothing in ADFUNN

Along the lines of the previous section, it is apparent that the learned functions are very well-regulated. It is possible, for a given set of analytical function prototypes to determine which analytical function matches best a given smoothed curve from the learned function output. Some points have never been adapted even though they are within the active range of adaptation, in which case they are in effect noise. To smooth these curves, we use the simple moving average method and least-squares polynomial smoothing, respectively.

4. Results

How do these smoothed curves work when substituted back into the neural network? We choose the simple moving average method, because it causes less distortion. For the natural language processing case, we substitute simplified curves smoothed by the simple moving average method for empirical ones for all of the 41 constituent tag output classes. The simplified curves work well, the correct classification is still 100% for all patterns used for training (254/254*100% = 100%) e.g. for the verb phrase neuron.

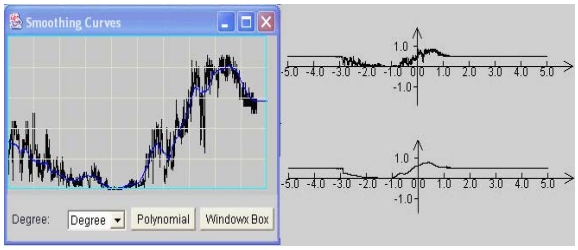

Fig. 4. Substituting the smoothed curve (the smooth line in the above left hand side) in the verb phrase neuron. The upper right hand curve is the original verb function output and the lower right hand curve is the smoothed version.

5. Related Work

Piazza et al [Piazza; Uncini] make use of an adaptive spline approach to function modification, and elsewhere both Fiori and Piazza [Piazza, Fiori] use a Digital Look-Up Table (LUT) for the activation function. ADFUNN is more general than the digital LUT approach, in the sense that it is an analogue algorithm incorporating linear piecewise interpolation between points. This analogue approach is effective for both sharp edges and smooth functions.

In the cubic analytic spline approach, a cubic level of complexity is assumed and a good suboptimal solution to the cubic spline curve fitting problem is applied. If the required function is linear, or a step, ramp or pulse function, splines are inappropriate; and in ADFUNN, we don't have the same problem of smoothness control that is faced by adaptive splines. Initial high precision allows for any f-shape, and subsequently, the function curve can be simplified by removing all points that have not been adapted, or where f is constant over a sub-range and by smoothing.

In addition, LUT requires a bounded input address and a suitable linear transformation (scaling and offset adding) has to be

performed on the output of the linear combiner in order to obtain the best LUT address; a problem not faced in ADFUNN which uses linear interpolation. Our method is essentially analogue and so hardware implementation is non trivial, but feasible using a combination of digital memory for F-points, amplifiers performing linear interpolation, and multiplier circuits.

6 Conclusion and Future Work

In this paper, we improved the general learning rule in ADFUNN and applied it to the Iris dataset, and a phrase recognition problem. Two function smoothing methods were compared for removing noise for the learned ADFUNN. Applying ADFUNN to some classification problems is highly effective even with no hidden nodes. Of the two smoothing methods, the simple moving average is more effective and computationally efficient than the least-squares polynomial smoothing. And the smoothed curves work well when substituted back into the ADFUNN neurons.

A secondary ADFUNN will be used to recognize the best matched analytical functions for substitution in the ADFUNN classifier. Intelligent data analysis with neural networks requires analysis of the weights to establish the most important factors and generate simplified equations to explain network decisions [Roadknight]. In ADFUNN, the learned functions also offer insights into the data [Palmer-Brown; Kang]. The development of algorithms for replacing learned functions with matched analytical ones to automatically generate mechanistic models from trained networks will be a significant contribution.

In more complex domains, such as with a much larger natural language dataset, there will be a need to use hidden nodes in a ML-ADFUNN. However, just as the single layer ADFUNN is more powerful than an SLP, so the multilayer ADFUNN is expected to be more powerful in learning than multi-layer perceptions (MLPs), and may well require fewer hidden neurons.

References

Fiori, S.: Hybrid Independent Component Analysis by Adaptive LUT Activation Function Neurons. Neural Networks, Vol. 15. (2002) 85-94

Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7(1936) 178-188

Garside, R., Leech, G. and Varadi, T.: Manual of Information to Accompany the Lancaster Parsed Corpus. Department of English, University of Oslo (1987)

Kang, M., Palmer-Brown, D.: An Adaptive Function Neural Network (ADFUNN) for Phrase Recognition. To appear in the International Joint Conference on Neural Networks (IJCNN), Montréal, Canada (2005)

Palmer-Brown, D., Kang, M.: ADFUNN: An Adaptive Function Neural Network. 7th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA), Coimbra, Portugal (2005)

Piazza, F., Uncini, A. and Zenobi, M.: Neural Networks with Digital LUT Activation Function. Proceedings of International Joint Conference on Neural Networks (IJCNN'93). Nagoya (Japan), Vol. 2. (1993) 1401-1404

Scheler, G.: Regulation of Neuromodulator Efficacy: Implications for Whole-Neuron

and Synaptic Plasticity. Progress in Neurobiology, Vol.72, No 6. (2004)

Scheler, G.: Memorization in a neural network with adjustable transfer function and conditional gating. Quantitative Biology, Vol. 1. (2004)

Roadknight, C.M., Balls, G., Mills, G. and Palmer-Brown, D.: Modelling Complex Environmental Data. IEEE Transactions on Neural Networks, Vol.8, No.4. (1997)856 – 862

Uncini, A., Piazza, F. and Vecci, L.: Learning and Approximation Capabilities of Adaptive Spline Activation Function Neural Networks. Neural Networks, Vol. 11, No. 2. (1998) 259-270