Hyperband networks: Description and automatic design

J. L. Crespo, Departamento de Matemática Aplicada y Ciencias de la Computa

Abstract

A new type of classification neural network is presented. The network consists of an 'hyperband' layer, a subset layer and the output layer. The first layer has standard continuous weights in its input connections, the second and output layers have no weights. A constructive algorithm is also presented that allows the network to be designed in order to learn a given data set without user interaction. The strategy for adjusting weights is compatible with many standard optimization algorithms. Examples of use and properties are discussed.

1 Introduction

This article presents a type of neural network focused on classification problems. Nevertheless, the output can have any meaning associated with an encoding in the $[0, 1]$ interval, not being restricted to a classification problem. For instance, each bit in the output may signal the membership to a given set, with these sets being possibly overlapping. Since this is a more general framework, we will keep the word *set* for the meaning of individual bits, and will implicitly admit, unless otherwise stated, the possibility of overlapping. Recent reviews of classification algorithms can be seen in [19] and [20].

For this type of network, we are going to see an automatic designing method. When building or designing a network, three approaches can be used:

- trial and error
- constructive, starting with a small or null network
- destructive, starting with a huge network

Constructive algorithms look more promising [3, 13, 12] than trial and error and destructive ones in order to find an acceptable network topology, small if possible; hence growing methods seem right candidates to solve the design problem.

Furthermore, if the task difficulty is to grow over time, a constructive growing method can evolve and solve the task by growing the network.

A review of growing methods for general problems can be found at [12]. Methods particularly suited for classification are surveyed in [13]; they usually have only one output positive for a given input, since they focus on the common problem of mutually excluding classes.

When following a constructive line, the learning process will be usually longer than that of a single network, because several networks of increasing size must be trained. Retraining weights has the advantage of a better location of global optimum network performance, and the disadvantage of an increasing learning time. This handicap can be alleviated if the number of parameters or weights to be adjusted at every step is small.

Although the most common approach is to train the whole set of weights in a neural network, in some methods, only a new hidden unit weights are trained (*cascade correlation* [6], *PPR* [8] or *GMDH* [14, 15] are examples for general networks, *upstart* [9] and *perceptron cascade* [5] are examples of binary networks). Due to the presence of weights in several layers in these networks, this means two small sets of weights to adjust for each incremental step: inputs to the new unit and new to output units.

Some other constructive algorithms retrain many weights at each step, or maybe just the new created ones and a few of the previously existing ones.

Now we will proceed to have a first description of the network in section 2, then we will get into deeper details and see an algorithm for designing it and a learning strategy in section 3. Next we will see some examples of applying this network with this algorithm in section 4, with a final discussion and conclusions in section 5.

2 Network presentation

The proposed network is based on the idea of combining *soft hyperbands*, which, in turn, are based on *hard hyperbands*. A *hard hyperband* is the intersection of two semispaces given by two parallel hyperplanes. Mathematically speaking, its definition is given by:

$$\{\mathbf{x_i}\}/|\mathbf{b} \cdot \mathbf{\hat{x}_i}| \leq 1$$

where:

$\mathbf{x_i}$: a point with dimension n

$\hat{\mathbf{x}}_\mathbf{i}$: the point expanded with a component valued 1, that is, with a dimension $n+1$

$\mathbf{b}$: set of parameters with dimension $n+1$ defining a hyperband

Then the center hyperplane can be given by:

$$\mathbf{b} \cdot \hat{\mathbf{x}} = 0$$

And the width is $\frac{1}{|\mathbf{b}'|}$, where $\mathbf{b}'$ is $\mathbf{b}$ without the expanded component, the one corresponding to 1 in $\mathbf{x}$.

A *soft hyperband* is a sort of fuzzy set associated to a hard hyperband.The value that controls the membership to the hyperband is the modulus:

$$|\mathbf{b} \cdot \hat{\mathbf{x}}_\mathbf{i}|$$

Membership decreases with this value. It is 1 for the center hyperplane and decreases with distance. In the proposed model the decrease is exponential. The membership function for any given point can be expressed as:

$$e^{-|\mathbf{b} \cdot \hat{\mathbf{x}}_\mathbf{i}|}$$

Since the network uses soft hyperbands, we will use the word hyperbands for these in the rest of the paper, unless otherwise noted. In the same vein we will simply call sets to the fuzzy or soft sets.

An intersection of hyperbands corresponds to regions where the minimum membership is still high. In this work we take the membership as the minimum of the membership of the individual hyperbands; this can be calculated from the maximum of the modulus calculated for each hyperband.

The final output is the union of several of these intersection sets. The union membership is taken in this work as the maximum of each set, so it is calculated from the minimum of the modulus of each set.

Then, the key features of the proposed network are:

- Use of the absolute value activation function in the *hyperband* units with dot product net input to build *hyperbands*
- Intersecting hyperbands, in the form of a maximum net input
- Output units joining these subsets, with a minimum net input function and a negative exponential activation function

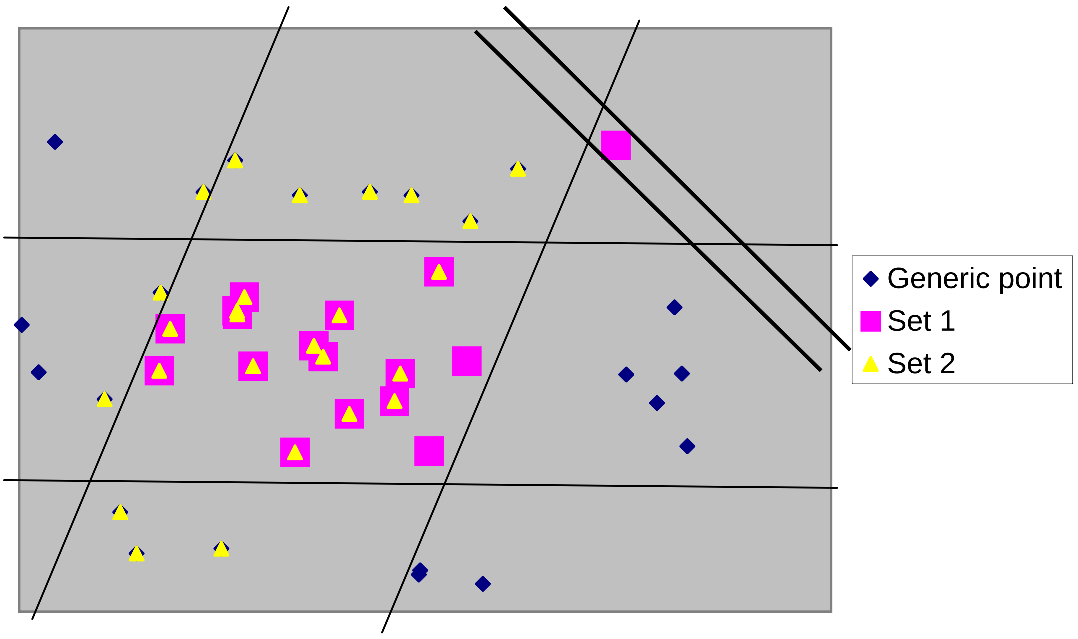

Figure 1: Example of a three set assignment problem

Interpretation of the network is then very simple, giving a collection of compact subsets for every set.

Take the example presented in fig. 1, where the hard hyperbands have been depicted in order to simplify the graphics. Set 1 is built with two subsets. The one in the center of the figure can be built with two bands, and the lonely point in the upper right with just one, always given the population present in the sample. Set 2 can be built with just one subset, consisting of a single band.

3 Algorithm and properties

Let us first describe in detail the static configuration, capabilities and properties of the proposed network and then we will proceed with the design and optimization method.

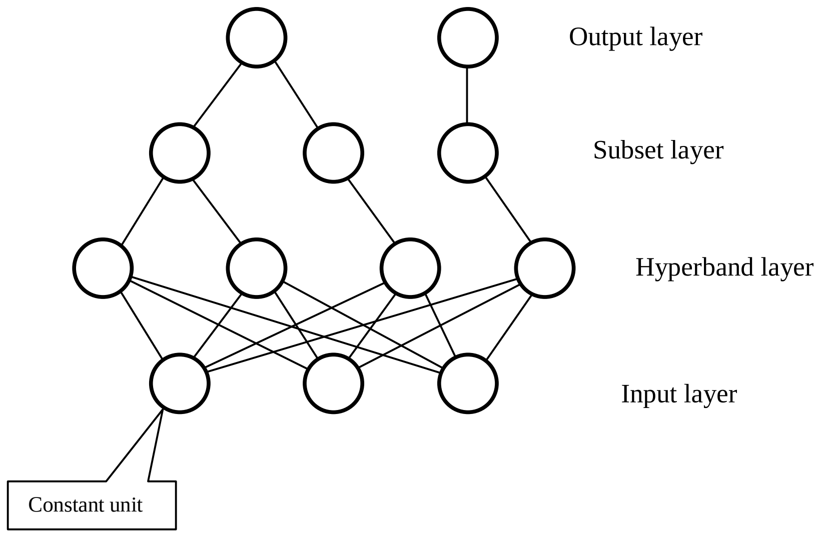

Figure 2: Network example with an architecture suited for the example in fig. 1

3.1 Static description

The network, an example of which is sketched in fig. 2, has the following layers, in order of loading or execution:

1. input
2. hyperband
3. subset
4. output

All processors in a given layer can execute in parallel.

The input layer has as many processors as input variables in the problem to be solved. No processing takes place in them. A constant input of one is added, for considering bias weights.

The hyperband layer consists of a variable number of processors fully connected from the input layer, with one input unit and one weight corresponding to each connection. The net input to each processor is the dot product of its weights and the input values. The output of each processor is the absolute value of the net input.

The subset layer has also a variable number of processors receiving connections from the hyperband layer. These connections have no weights. A given processor in this layer receives its input from several, not all, of the processors in the hyperband layer, and produces as output the maximum of its inputs.

The output layer has as many processors as outputs in the problem (it may vary in the sense that new outputs may be added). Each output unit receives one or several connections from units in the subset layer, with no weights; the net input is the minimum of its inputs and the activation function is the negative exponential.

Summarizing, let us describe the algorithm for network execution:

3.1.1 Executing the network

Allowing for all processors in the same layer to work in parallel,

1. take input values
2. perform dot product (input $\cdot$ weights) at each hyperband processor
3. apply absolute value at each hyperband processor
4. take linked data from the hyperband to the subset layer
5. calculate maximum at each subset processor
6. take linked data from the subset to the output layer
7. calculate minimum at each output processor
8. apply negative exponential at each output processor

As already mentioned, the network only has weights in the input to hyperband connections. This reduces the learning time and, therefore, evolving, or designing, the network.

3.2 Capability

We will see here a qualitative overview of the network representation power.

Each neuron in the hyperband layer identifies a hyperband in the input space, that is, the subspace comprised between two parallel hyperplanes. The hyperband position is indicated by the weight vector direction, and the width, or distance between hyperplanes, is given by the weight vector modulus; this way a nearly null weight vector corresponds to extremely wide hyperbands, and, in the limit, a null weight vector comprises all input space.

If we compare hard hyperband units with the threshold ones (step activation function) commonly used in Boolean networks, we see that, for bounded inputs, a hyperband has at least the same identification power than a semispace, which is the output of threshold units. We only have to keep one of the hyperband borders at the same position that the step border and the other border of the hyperband outside the bounded set. An analog reasoning can be made at comparing soft hyperbands with logistic units.

In a n-dimensional input space, such as R^n, any subset of points that is limited with hyperplanes can be obtained with the intersection of hyperbands, at least n if we want the subset to be finite. This intersection is the output of a unit in the subset layer. Then, by choosing arbitrarily narrow hyperbands and placing then so they intersect, we can build arbitrarily small polyhedral boxes around any point.

Furthermore, any finite bounded set can be built as the union of compact subsets, and this union is the output of the output layer. Therefore, the network will be able to produce quite general mappings.

3.3 Automatic design

Now, it is turn to face the network construction. Let us proceed to the description of a network design and training process in a top-down way. We will start with the top-level description of the process:

3.3.1 One possible top-level strategy to building and training the network

1. Iterate for each output until the final training error average varies less than a given threshold:
   (a) Extract a random subset for training and the rest for testing
   (b) Build a network reducing the training error until the test error starts increasing
   (c) Update the final training error average
2. Take the whole sample for training
3. Build a network reducing the training error until it reaches the training error average obtained before

3.3.2 Significant features of this top-level construction method

- Overfitting is prevented by measuring repeatedly when test error starts increased
- The method implies several building phases, so these should be fast, for it to be tractable

3.3.3 One possible "build-a-network" strategy

Now, let us enter the "*Build a network*" step description For each output: Iterate adding a subset processor and its hyperband processors, fitting one of these at a time

3.3.4 Significant features of this construction method

- No more than one hyperband processor is trained each time; this means easier learning
- It is possible to check the usefulness of previous hyperbands or subsets, but this can lead to suboptimal performance
- In order to make sure that each hyperband processor helps reduce the error rate, the starting weight point is not random but is set to include the first positive output failed for the first hyperband of the subset and to exclude the first negative output failed for subsequent processors

The fitting algorithm for each individual hyperband processor can be any gradient-directed algorithm, like quasi-newton, Levenberg-Marquardt, conjugate gradient or the like.

3.4 Objective function

The above process is valid for any objective function. We describe here a variation of the usual Minimum Square Error that has the purpose of helping reduce the class balance problem. Let us define the two types of errors that the network can produce:

- Type A: output 1 where it should be 0
- Type B: output 0 where it should be 1

With this in mind, let us use the following notation:

r_i: the model answer at point i

c_i: the right answer at point i

N_A: number of data points in the current sample that are outside the set being trained (the answer should be 0 for the corresponding output unit)

N_B: number of data points in the current sample that belong to the set being trained (the answer should be 1 for the corresponding output unit)

The proposed objective function, E, is made of two components:

- Type A errors:

  $$E_\mathrm{A} = \begin{cases} 0 \text{ if } N_\mathrm{A} = 0 \\ \frac{\sum (r_i - c_i)^2}{N_\mathrm{A}} \text{ in other case} \end{cases}$$

  This is proportional to the number of incorrectly accepted inputs and should obviously be reduced

- Type B errors:

  $$E_\mathrm{B} = \begin{cases} 0 \text{ if } N_\mathrm{B} = 0 \\ \frac{\sum (r_i - c_i)^2}{N_\mathrm{B}} \text{ in other case} \end{cases}$$

  This is proportional to the number of incorrectly rejected inputs and should also be reduced

Then, the total objective function is simply the addition of these two terms :

$$E = E_\mathrm{B} + E_\mathrm{A}$$

The way each error type affects the total function is independent of the number of points. The effect of this can be seen if we consider a sample with a lot of non-set points and a few set points; let us calculate the error of a unit rejecting all points with a lumped error function like ordinary MSE:

$$E_l = \frac{\sum (r_i - c_i)^2}{N} \sim 0$$

With the proposed function the result would be:

$$E = 0 + 1 = 1$$

We see that this is flagged as a bad solution with the proposed objective function, hence this is aimed at requiring good performance equally for false positives and negatives.

4 Examples

In this section some examples of applications of the model are presented so its strengths and weaknesses can be analyzed. We will use some common examples already tested with many methods in the literature.

We are going to see the results of it in several common benchmarks: the iris [7], breast cancer[1] [16], soybean disease [17] and animal class [18] identification problems, all of them available at the UCI repository database, [4].

We follow the standard 10 cross-validation procedure, but since randomness affects initialization we repeat the whole process until the performance average varies less than 1%. Quasi-Newton optimization is used for hyperband weights training.

Table 1 summarizes the features and results obtained in each problem comparing them with other average results published in review papers.

In this table we can see that hyperband networks have good performance in some problems (iris, cancer), average in other problems (soybean) and low in other (zoo). Concerning the number of parameters, although the hyperband network has weights only at the hyperband layer, the larger amount of hyperbands needed to represent a problem compensates for it. We can see that training times (which include the design of the network) are rather low.

5 Discussion

We have addressed two issues:

1. A particular type of network
2. A build-up algorithm for this network

Concerning the presented network model, it has as distinct features that indicate its possible use:

1. Only the input to hyperband connections have adjustable weights
2. It uses absolute value, minimum, maximum and negative exponential functions

The first feature makes learning simpler. The second feature is a particular choice; different networks may be pursued by changing the intersection and

[1]Originally obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg

Problem	Iris flower	Breast cancer	Soybean disease	Animals
Number of inputs	4	9	35	16
Number of classes	3	2	19	7
Number of samples	150	698	683	100
Naïve Bayes average accuracy in [19]	96	96	93	95
C4.5 average accuracy in [19]	95	95	92	93
3 nearest neighbors average accuracy in [19]	95	97	91	93
Backpropagation neural network average accuracy in [19]	85	96	93	60
SMO average accuracy in [19]	85	97	93	93
SMO average accuracy in [20]	92	n/a	86	78
C4.5 average accuracy in [20]	94	n/a	88	89
Backpropagation neural network average accuracy in [20]	78	n/a	81	76
k nearest neighbors average accuracy in [20]	96	n/a	83	92
Hyperband network average accuracy	97	97	90	81
Average number of weights of hyperband network	36.5	32.5	889.2	166.6
Time to build a hyperband network	0.44 s	1.16 s	2 m 26 s	1.46 s

Table 1: Results summary and comparison with average results obtained with other classifiers. The accuracy is reported in percentage. The timing was measured in a Pentium 4 at 2.8 GHz with an 800 MHz bus

union definition, as it is made in fuzzy set theory, and choosing a different membership function.

The particular structure of the model biases it for data that is best represented by the proposed combination of functions. Nevertheless the accuracy seems to be comparable to other common classification methods.

Concerning the constructive algorithm, its key features are:

1. It is aimed to reduce the generalization error; this can be substituted for other regularization method

2. It adjusts only one processor's weights each time

3. It accounts separately for the different type of errors, rescaling them, so they have the same relative importance; other objective error functions can also be used

4. It is compatible with any gradient-based learning algorithm, hence allowing for fast optimization

The combination of these features allows for a rather effective design in a small computer time. Considering the results we can conclude that the strategy works.

The author wishes to express his sincere gratitude to Reyes Ruiz for her support, comments, ideas and corrections.

ϵ

References

[1] F. Aluffi-Pentini, V. Parisi and F. Zirilli, Algorithm sigma, a stochastic integration global minimization algorithm. A.C.M. Transactions On Mathematical Software , vol. 14, no. 4 , pp. 366-388.

[2] T. M. Apostol. (1974) Mathematical analysis. Addison-Wesley, Reading.

[3] T. Ash and G. Cottrell, (1998) Topology-Modifying Neural Network Algorithms. In: M. A. Arbib, (ed) The handbook of brain theory and neural networks, The MIT Press.

[4] C.L. Blake and C.J. Merz. (1998) UCI Repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. Available http://www.ics.uci.edu/ mlearn/MLRepository.html

[5] N. Burgess, (1994) A Constructive Algorithm That Converges for Real-Valued Input Patterns, International Journal of Neural Systems, 5(1), pp. 59–66.

[6] S. E. Fahlman and C. Lebiere, The cascade-correlation learning architecture. In: D. S. Touretzky, (ed) Advances in Neural Information Processing Systems, 2, Morgan Kauffmann, San Mateo, CA, pp. 524–532.

[7] R.A. Fisher, (1936) The use of multiple measurements in taxonomic problems. Annual Eugenics, 7, Part II, pp. 179-188.

[8] T. E. Flick, L. K. Jones, R. G. Priest and C. Herman, (1990) Pattern classification using projection pursuit., Pattern Recognition, vol. 23, no. 12, pp. 1367–1376.

[9] M. Frean, (1990) The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks. Neural Computation, 2, pp. 198–209.

[10] R. J. Henery (1994) Classification. In: D. Michie, D. J. Spiegelhalter and C. C. Taylor, (eds) Machine Learning, Neural and Statistical Classification, Ellis Horwood.

[11] Y. Ichikawa and T. Sawa, (1992) Neural network application for direct feedback controllers. IEEE Transactions on Neural Networks, v. 3, n. 2.

[12] T.-Y. Kwok and D.-Y. Yeung, (1997) Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems. IEEE Transactions on Neural Networks, vol. 8, no. 3, pp. 630–645.

[13] R. Parekh, J. Yang and V. Honavar, (2000) Constructive neural-network learning algorithms for pattern classification, IEEE Transactions on Neural Networks, vol.11, no.2 , pp.436–51.

[14] R. E. Parker and M. Tummala, (1992) Identification of Volterra systems with a polynomial neural network. In: Proc. 1992 IEEE Int. Conf. Acoust., Speech, Signal Processing, San Francisco, CA, vol. 4, pp. 561–564.

[15] M. F. Tenorio and W. T. Lee, (1990) Self-organizing network for optimum supervised learning. IEEE Transactions on Neural Networks, vol. 1, pp. 100–110.

[16] O. L. Mangasarian and W. H. Wolberg, (1990) Cancer diagnosis via linear programming. SIAM News, Volume 23, Number 5, pp 1–18.

[17] R.S. Michalski and R.L. Chilausky, (1980) Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis. International Journal of Policy Analysis and Information Systems, Vol. 4, No. 2.

[18] Richard Forsyth. PC/BEAGLE User's Guide

[19] S. B. Kotsiantes et al. (2006) Machine learning: a review of classication and combining techniques. Artif Intell Rev, 26, pp. 159-190

[20] Niklas Lavesson and Paul Davidsson, (2006) Quantifying the Impact of Learning Algorithm Parameter Tuning. The Twenty-First National Conference on Artificial Intelligence (AAAI-06)