Radial Basis Function Neural Networks with Example Weights and LMS Linear Regression Weight Settings

Jiřina, Marcel
2001

Institute of Computer Science
Academy of Sciences of the Czech Republic

Radial Basis Function Neural Network with Example Weights and LMS Linear Regression Weight Setting

M. Jiřina

Technical report No. 848

November, 2001

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: (+420)6605 3350, fax: (+4202) 858 57 89,
e-mail:marcel@cs.cas.cz

Institute of Computer Science
Academy of Sciences of the Czech Republic

Radial Basis Function Neural Network with Example Weights and LMS Linear Regression Weight Setting

M. Jiřina

Technical report No. 848

November, 2001

Abstract:

This report solves the problem, which arises in tasks where the importance or weight of individual learning examples is not equal and is known in advance. A single hidden layer RBF neural net is described which allows considering different example weights in the learning process. This is made in two ways. First, when computing settings of centers by cluster analysis, the distance of learning examples or centers is defined so, that it is the larger, the smaller is the weight of one or another example. The second, we use weighted LMS linear regression technique for setting of connection weights of connections from centers in the hidden layer to the output neuron or neurons.

Keywords:
RBF neural network, weight of learning examples, weighted linear regression, weighted distance of examples.

Radial Basis Function Neural Network with Example Weights and LMS Linear Regression Weight Setting.

Abstract

This report solves the problem, which arises in tasks where the importance or weight of individual learning examples is not equal and is known in advance. A single hidden layer RBF neural net is described which allows considering different example weights in the learning process. This is made in two ways. First, when computing settings of centers by cluster analysis, the distance of learning examples or centers is defined so, that it is the larger, the smaller is the weight of one or another example. The second, we use weighted LMS linear regression technique for setting of connection weights of connections from centers in the hidden layer to the output neuron or neurons.

Keywords

RBF neural network, weight of learning examples, weighted linear regression, weighted distance of examples.

Contents

Acknowledgment

This work was supported by the Ministry of Education of the Czech Republic under Project No. LN00B096.

Radial Basis Function Neural Network with Example Weights

1. Introduction

One of problems which can be encountered with neural network learnning is the fact, that not all examples presented have the same weight with respect to their influence or significance in the learning set. This problem can be found in some problems with nonequal but known (measurable) quality of data or where the significance is given by some variable known in advance. In particle physics this variable can be so called effective crossection of a particle corresponding to particular example in the learning data set [8]. Moreover, in the testing phase one prefers to have the results more reliable for examples with high weight than for examples with lesser one. We then differentiate between connection weight (i.e. transfer ability of a connection) and the example weight, i.e. significance of a given example or sample of the training set.

This of different example significance, i.e. weight, can be relatively easily solved by presenting the examples with higher weight more often than the others. It supposes large amount of learning iterations to make it possible. In systems based on some kind of optimization such an approach has a limited use. In the RBF network we use linear regression for optimization instead of iterative process. The example weights are included in regression equations as well as into centers selection by cluster analysis.

The program is written in ANSI C++ for use under DOS/Windows as well as under UNIX/Linux environment without any modification. Under DOS/Windows environment it was written under Pacific C system [6] and Borland C++ system [7].

This type of neural network was used for comparison of GMDH and NNSU abilities with “standard” type of neural network. Some results and comparisons are shown.

2. Basic features of RBF NN

Radial basis function neural networks are well known in literature and they are also a part of different tools and application programs [1], [2], [3].

In our case the RBF neural net has three layers - one input layer, hidden layer, and output layer, see Fig. 1.

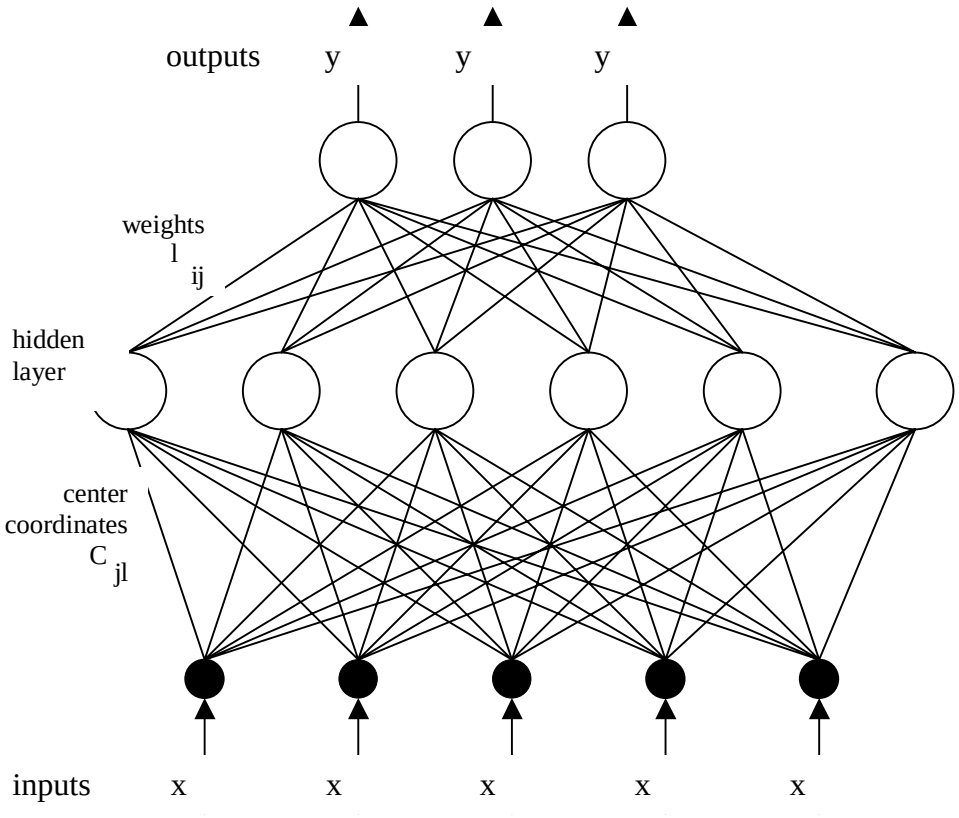

Fig. 1 Single hidden layer RBF neural network.

The neurons in hidden layer have RBF transfer function. There are *n* inputs, *m* outputs and *k* hidden neurons. Each neuron of the hidden layer is connected to all inputs and all neurons in the hidden layer is connected to the output neuron or neurons in case of several outputs. RBF NN belong to feed-forward neural networks as in recall phase the signal simply goes one way from inputs to outputs.

Let $x=(x_1, x_2, \ldots x_n)$ be vector of inputs,

$y=(y_1, y_2, \ldots y_m)$ be vector of outputs.

The input-output transformation is given by equations

$$y_i = \sum_{j=1}^{k} {}_{ij}\, f(s_j, \|x-C_j\|), \quad i=1, 2, .. m, \qquad (1)$$

where ${}_{ij}$ are output layer weights, $Cj=(C_{j1}, C_{j2}, \ldots C_{jk})$ is a center of RBF function, s_j is a parameter of particular RBF function expressing how broad the RBF function is, and *f* is a radial basis function as follows. In our case we use ($s = s_j$, $x = \|x-C_j\|$) - see Fig. 1

Gaussian RBF $f = \exp(-(x/s)^2)$

Spline RBF
$$f = \frac{1}{4x^2 \log \frac{|x|}{s^2|s|+1} + 1}$$

Bell function RBF $f = \frac{s}{\sqrt{4x^2 + s^2}}$.

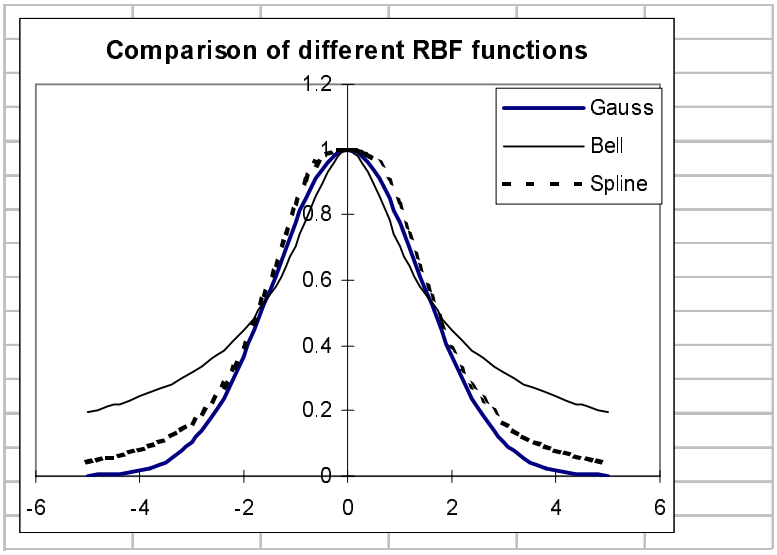

Fig. 2 Different RBF functions.

3. RBF NN Learning

Basic steps

Learning of RBF NN consist of three steps:

1. Stating of centers C_j for all hidden neurons. There are two methods, random selection from the learning set, and k-means [1] (Jancey [9]) method of cluster analysis. We do not describe them in detail.
2. Stating of RBF function broadness parameter s_j (or simply s above). The broadness parameter s_j is an average distance of q nearest neighbours from the centre C_j.
   - Concretely, initially set $s_j = 0$. For each hidden neuron do the following:
   - To get q nearest neighbours compute for all samples in the training set the $D=\|w_{ij}-x_{ik}\|$.
   - If (D>) then there is a different sample than particular center C_j.
   - If the sample is already a member of the set of q nearest neighbours then ommit it, else add D to s_j. Finally s_j is equal to sum of distances of center C_j to q nearest neighbours from the learning set. In the end set $s_j=s_j/q$ to get average which is considered as broadness parameter of RBF function with the center C_j.
1. Computation of weights $_{ij}$
   - either by iterative adaptation, ie. by standard often used procedure: For it we use simple iterative procedure controlled by learning parameter :
     - $_{ijr} =_{(ijr-1)} + (d_j - y_j)s_j$

- where d_j is a desired value of output No. j and y_j is RBF NN response to one sample from the learning set. At the same time a cumulated error is evaluated according to formula
- error(r)=error(r-1) + $|d_j-y_j|$.
- The process is stopped if accumulated error reaches a given limit or there was performed a given number of iterations. Of course, the process one can stop manually; in any case the result of learning can be as RBF NN file.

- or by linear regression: The iterative optimization procedure for weights $_{ij}$ of one output neuron can be substituted with least squares minimization by linear regression using well known general formula $(X^tX)^{-1}X^td$, where ^{t} denotes transposition, $d=(d_1, d_2, ... d_k)^t$ is column vector of desired values of the learning set, and X is matrix of hiddden neurons outputs (columns) for all samples (rows) of the learning set. This approach would eliminate influence of learning parameter on speed of learning and is explained in detail in corresponding chapter.

Distance using example weights

Positions of RBF centers can be chosen either by random selection or by cluster analysis.

If the number of centers *m* is given (or already selected) then in the first case one simply chooses *m* different patterns, examples from the learning set. To consider example weights the procedure can take into account the example weights so that the examples of higher weight have larger probability to be chosen as a centre of RBF.

The other way for RBF centers selection is a cluster analysis. We use *k*-means [1] (Jancey [9]) algorithm with fixed in advance given number of clusters. Essential notion for cluster analysis is notion of distance. We try to consider different example weights by method very similar to gravitation law or Coulumb's law

$$F = c\frac{m_1 m_2}{r^2},$$

where *F* is force, *r* distance of two bodies, and m_1, m_2 their masses or electric charges in case of Couloumb's law, *c* is a constant. Let us recompute the distance to value *R* correponding to bodies with unit mass or charge

$$c\frac{m_1 m_2}{r^2} = c\frac{1}{R^2}.$$

From it follows

$$R = \frac{r}{\sqrt{m_1 m_2}}.$$

We use this formula for computation sample distance if sample weights are given. In that case m1, m2 are sample weights. These weights need not be normalized and then we use form

$$R = r\frac{\overline{m}}{\sqrt{m_1 m_2}}.$$

$\overline{m}$ denotes mean value of all example weights and which becomes $R = r$ for all weights equal.

Linear regression with weights

The weights of connections from the hidden layer to the output neuron one can very effectively adapt using mean square error minimization by linear regression. At the same time, it is possible to take into account the example weights as mentioned above.

This is only to remind standard linear regression with weights of individual cases, examples or rows of data matrix.

Let data be given in form of individual examples each in one row:

$$\begin{array}{cccccc} x_{11} & x_{12} & \ldots & x_{1n} & w_1 & d_1 \\ x_{21} & x_{22} & \ldots & x_{2n} & w_2 & d_2 \\ & & \ldots\ldots\ldots & & & \\ x_{k1} & x_{k2} & \ldots & x_{kn} & w_k & d_k \end{array}$$

total k examples be given. x_{ij} (i = 1, 2, ... n) are input values, d_j the desired output and w_j the example weight. The matrix above can be written in form

$$X_2 W d.$$

For least squares linear approximation with coefficients $b = (b_0, b_1, \ldots b_n)$ let us construct a matrix $X = 1X_2$, where 1 denotes one column of 1's more left from the matrix X_2 (to include a constant) and then standard solution without considering weights is

$$b = (X^t X)^{-1} X^t d$$

and with weights

$$b = (\hat{X}^t X)^{-1} \hat{X}^t d ,$$

where $\hat{X}$ denotes matrix X with all elements in each row multiplied by corresponding example weight w_i . Of course the equations above are not solved using matrix inversion but as a system of linear equations.

4. A short program handbook

The progrm consist of two parts, part for learning neural network, in fact for genertng of learned RBF NN as a file, and a part for usage of the learned RBF NN as a function approximator according to eq. (1). The both programs are controlled by command line parameters.

The learning program RBF-LRNW.exe

Usage: RBF-LRNW.exe <learning set file name> <No. of inputs> <No. of outputs> [parameters]

where

<learning set file name> in this file each column represents one input variable, each row is

	one learning example. The file should contain already normalized data but there is an option alowing to use nonnormalized data as well,
<No. of inputs>	Number of first n columns representing input variables,
<No. of outputs>	Number of next m columns representing output variables,
[parameters]	Need not be used, then implicit values are used;
s=xxx	Number of neurons in the hidden layer (impl. s=15);
f=g|s|b|z	Transfer function - Gaussian, Spline, Bell (b or z) (impl. f=g);
q=xxx	No. of nearest neighbours for broadness computation q<s (impl. q=1);
i=xxx | r=xxx	No. of iterations or maximal error (impl. i=1000); when needed early, hit any key to break iterations; if parameter *a* is set the net will be saved in a file even if manual break occurs; when linear regression is used no manual break is possible;
i=0 | r=0	Noniterative computation using LINEAR REGRESSION;
n=t|f	Normalize the learning data (True|False) to zero mean and unit dispersion (impl. n=f);
e=xxx	Learning parameter (eta) (impl. e=0.01);
p=xxx	Show results of iterations after each xxx steps (impl. p=100, max. 5000)
z	Do not stop when displaying current results;
c=n|s	Selection of centers: randomly without repetition(n) from the learning set file, cluster analysis of the learning set file (s) (impl. c=n)
a=NNname.net	Save the resulting neural net to the file NNname.net (impl. don't save).

Initially the weights lambda are set to random values <-0.1;0.1>.

The RBF function approximator RBF-TSTW.exe

Usage: RBF-TSTW.exe <RBF net file name> <testing set file name> [parameters]

<RBF net file name>	is name of already learned RBF neural net;
<testing set file name>	Each column of this file represents one input variable, each row is one example; number of inputs is given by neural net file and they correspond to first n columns of the testing set file. In the testing set file may follow next m columns corresponding to output values.
[Parameters]	(mandatory if corresponding values are in the data file)
t=xxx	number of outputs; if there are output values in the testing set file this parameter is mandatory, otherwise it is not used.

w	in the data file are given the example weights (mandatory if these values are in the testing set file)

Output of this program is file results.res containig the same number of data rows as the testing set file. In results.res after columns of original testing set file (including optionally weights and desired output values) follows mark => and then the output values generated by RBF NN.

5. Results

The approach used in the RBF method and program presented tries to diminish the largest disadvantage of RBF as well as backpropgation algorithm in necessity to estimate properly the learning parameter, to choose most suitable RBF function, and choose size of neighbourhood by stating number of its members q. From these prameters the learning parameter is most essential for speed of learning. This disadvantage is solved by excluding parameters of iterative procedure by using linear regression (no parameters to set) and by standardized settings. The nonstandard settings are only for user's experimentation and to keep some compatibility with possibilities of standard approaches.

We found that for standard and double arithmetics it is possible to use up to 120-150 hidden neurons before numerical problem in linear regression occurs. As to learnind data size, 15000 items of 23 input values (inputs) caused no problems.

6. Future plans

Future modification of this program will use other kinds of RBF function more, or even not exactly RBF functions, namely functions mentioned e.g. in [10].

There exist lot of other variants and modifications of RBF neural networks, namely a version with iterative adaptation of centers and weights at the same time [1]. The method cited does not accounts different sample weights and we hope to develop its variant with sample weights.

A rather different approach are the growing networks reminding GMDH neural network principle [4]. The building a neural network starts with a single neuron and according to error evaluation new neuron is connected and learned without modification of settings of already existing neurons [11].

As to the problem of overtraining, it may occur but it can be solved the same way as in GMDH [4], [5] method - by splitting learning set to training and to evaluating part.

References

[1] Schwenker, F., Kestler, H. A., Palm, G.: A Comparison of LVQ and RBF Networks for Classification and Data Clustering. 6th Microcomputer School, Neural Networks Theory and Application, Brno, Czech Republic, CCB, s.r.o., 1994, pp. 233-238.

[2] Orr, M. J. L.: Introduction to Radial Basis Function Networks. April 1996, http://www.anc.ed.ac.uk/~mjo/intro/intro.html

[3] NeuralWorks Professional II/PLUS, NeuralWare, Inc., Pittsburgh, Penn., USA, 1991

[4] Ivakhnenko, A.G.: Polynomial Theory of Complex Systems. IEEE Trans. on System, Man and Cybernetics, SMC-1(1971), No. 4, pp. 364-378

[5] Tamura, H., Kondo, T.: Heuristics free group method data handling algorithm of general optimal partial polynomials with application to air pollution prediction. Int. J. Systems Sci., vol 11 (1980), No. 9, pp. 1095-1111.

[6] Pacific C Programmers Development Environment for MS-DOS V7.51, Freeware Version. (C) 1984-2000 HI-TECH Software. PO Box 103, Alderley QLD 4051 Australia, hitech@htsoft.com, http://www.htsoft.com, ftp.htsoft.com.

[7] Borland C++, Borland International, Inc.

[8] ATLAS Technical Proposal for a General-Purpose pp Experiment at the Large Hadron Collider at CERN. CERN/LHCC/94-43, LHCC/P2, 15 December 1994.

[9] Lukasová, A., Šarmanová, J.: Cluster analysis methods (in Czech), SNTL Praha, 1985.

[10] Sosík, P.: RBF Networks with Quasi-interpolating Functions. 6th Microcomputer School, Neural Networks Theory and Application, Brno, Czech Republic, CCB, s.r.o., 1994, pp. 151-157.

[11] Esposito, A., Marinaro, M., Oricchio, D., Sarpeta, S.: Approximation of Continuous and Discontinuous Mappings by a Growing Neural RBF-based Algorithm. Neural networks (Pergamon Press) vol. 13 (2000), pp. 651-665.

***

**

*