Learning Heterogeneous Functions from Sparse and Non-Uniform Samples

Dragoljub Pokrajac and Zoran Obradovic
School of Electrical Engineering and Computer Science
Washington State University, Pullman, WA 99164-2752, USA
{dpokraja, zoran}@eecs.wsu.edu

Abstract. *A boosting-based method for centers placement in radial basis function networks* (*RBFN*) *is proposed. Also, the influence of several methods for drawing random samples on the accuracy of RBFN is examined. The new method is compared to trivial, linear and non-linear regressors including the multilayer Perceptron and alternative RBFN learning algorithms and its advantages are demonstrated for learning heterogeneous functions from sparse and non-uniform samples.*

1. Introduction

Target functions for regression in real-life situations are often highly heterogeneous, while samples are sparse and collected from a non-uniform distribution. For example, this is typically the case in spatial domains like oceanography or precision agriculture where sampling resolution varies with locations [Pennington and Volstad, 1991]. Obviously, prediction quality in such situations cannot be too high, but there is still a need for constructing non-linear models that are better than trivial and linear predictors. Although it was observed that learning in such environments is substantially different from learning when high accuracy is achievable [Abu-Mostafa, 1995], it appears that non-linear regression for heterogeneous functions from sparse and non-uniform samples has not been studied much by the neural networks community.

Multilayer Perceptron networks [Haykin, 1999] tend to loose their generalization ability with heterogeneous, sparsely sampled data. This is due to their neurons providing a global discrimination, which results in models generalizing even in regions of feature space where there was no training data. In contrast, radial-basis-function networks (RBFN) with Gaussian neurons [Poggio and Girosi, 1990] provide local approximations, and thus are potentially more suitable for estimating non-homogeneous functions by avoiding improper generalization. However, learning algorithms for RBFN are, in general, developed assuming that a fairly large data set is available for training [Carse and Fogarty, 1996; Chen, 1995; Cherkassky and Najafi, 1991; Fritzke, 1994; Orr at al, 1999; Saha, Christian and Tang, 1989].

To address learning heterogeneous functions from sparse and non-uniform samples, a new RBFN method based on boosting regressors [Drucker, 1997] applied to radial basis centers is proposed in Section 2. The algorithm accuracy as well as the influence of sampling techniques in boosting is examined in Section 3.

2. Method

Boosting is a relatively new technique aimed to reducing modeling error by increasing a prediction margin [Freund and Schapire, 1996; Schapire et al, 1998]. It is based on successive data re-sampling according to the accuracy of previously generated models. Learning in boosting occurs in iterations. At each iteration, an embedded learning model is trained on a data subset obtained according to adaptive distribution probabilities. Each trained model is evaluated on the entire training data, and distribution probabilities are updated such that samples predicted with higher accuracy are less likely to be sampled at the next iteration. This results in a sequence of models that gradually explore particular regions of a function domain. The final model is obtained as a combination of all trained models weighted according to their accuracy on the entire training data.

Although originally developed for classification, boosting was recently generalized to regression, and successfully employed with regression trees and multilayer Perceptrons [Drucker, 1997; 1999]. In the following sections, in addition to a direct approach to boost RBFN models by adapting data sampling distribution probabilities, we propose its modification with boosting radial-basis function centers.

2.1 Boosting RBFN

The proposed boosting-based algorithm for RBFN centers placement, denoted as AdaBoostC, is shown in Fig. 1. Similar as in AdaBoostR [Drucker, 1997], linear, quadratic and exponential loss functions are considered. In Fig.1, loss L_i is defined as $|y_i - \hat{y}_t(\mathbf{x}_i)|/d$ in linear, $|y_i - \hat{y}_t(\mathbf{x}_i)|^2/d^2$ in quadratic and $\exp(-|y_i - \hat{y}_t(\mathbf{x}_i)|/d)$ in exponential case (where $d = \max(y_i - \hat{y}_t(\mathbf{x}_i))$, while y_i and $\hat{y}_t(\mathbf{x}_i)$ are a true target value corresponding to $\boldsymbol{x}_i$ and its estimate using t-th predictor designed in a boosting sequence). However, AdaBoostC differs from AdaBoostR in the following:

- in AdaBoostC, models are constructed on all training data, while in AdaBoostR samples are drawn according to a probability distribution;
- a probability distribution is applied to a radial basis centers assignment in AdaBoostC;
- the mean of trained models outputs (simple or weighted by log $1/\beta_t$) is also considered for generating the final model in AdaBoostC, in contrast to using only the weighted median in AdaBoostR;
- several methods for drawing training samples are considered in AdaBoostC (a roulette wheel, stochastic remainder with and without replacement, and deterministic sampling [Goldberg, 1989]).

Another way to apply AdaBoostR to boost RBFN is to perturb training data. As before, the four drawing algorithms and the final model integration using mean-based heuristics from AdaBoostC are viable possibilities.

2.2 Alternative RBFN training algorithms

To compare AdaBoostC and AdaBoostRBFN with other popular learning algorithms for RBFN, we consider an initialization of RBFN center positions using self-organized maps (RBFN-SOM) [Kohonen, 1984] and regression trees (RBFN-RT) [Orr at al, 1999], as representatives of unsupervised and supervised methods, respectively. Both algorithms are also used as embedded learners in AdaBoostRBFN and these variants are called AdaBoostSOM and AdaBoostRT. In addition, we considered RandomC where RBFN centers are randomly chosen (without replacement) from the training set [Lowe, 1989].

RBFN-SOM, AdaBoostSOM and RandomC require the number of centers N_c and the radial-basis function shape to be pre-specified. For simplicity, we worked with symmetric radial-basis functions (the same prespecified radius r for all independent variables). For RBFN-RT and AdaBoostRT the number of centers and radii are automatically determined. Our intention was to show potential advantages and drawbacks of the considered algorithms, and therefore their parameters were not optimized.

2.3 Test data

Artificially generated data were used in order to perform fully controllable experiments. Typical regression benchmarks are fairly homogeneous slow changing functions, where RBFN can achieve high accuracy using a relatively small training sample size [Friedman, 1991; Orr at al, 1999; Drucker, 1999]. Such data is inadequate for testing the capabilities of RBFN algorithms to learn heterogeneous, fast changing functions. Therefore, our experiments were performed using data generated by

$$y = \sin a_i \pi x_1 \cos b_i \pi x_2 \tag{1}$$

where $i \in \{1,2,3,4\}$ corresponds to the quadrant to which $\mathbf{x}=(x_1,x_2) \in [-1,1]^2$ belongs. Here, the function shape is determined by the coefficients a_i, b_i. For $a_1=\ldots=a_4$ and $b_1=\ldots=b_4$, equation (1) models homogeneous periodic functions which are a generalization of a test function used in [Orr at al, 1999]. Opposite, by choosing distinct coefficient values in different quadrants, desired levels of function heterogeneity can be modeled.

Training:
input:
- sequence of examples $(\mathbf{x}_i,y_i)$, $i=1,\ldots,N_{train}$;
- the number of RBFN centers N_c;
- an algorithm RBF_LEARN for learning RBFN;
- maximal number of iterations T;

initialize probability distributions: $D_t(i)=1/N_{train}$, for all i;
initialize $t=1$; $L_{avg}=0$;
while average loss $L_{avg}<0.5$ and $t \le T$
- randomly draw N_c RBFN centers according to D_t;
- train model $h_t: \mathbf{x} \rightarrow y$ with RBF_LEARN using all $(\mathbf{x}_i,y_i)$;
- calculate linear, quadratic or exponential loss L_i;
- calculate weighted average loss $L_{avg}=\Sigma_i (D_t(i)L_i)$;
- set $\beta_t=L_{avg}/(1-L_{avg})$;
- update $D_{t+1}(i) = D_t(i)\beta_t^{1-L_i}$;
- normalize $D_{t+1}= D_{t+1} / Z_{t+1}$ to obtain a probability distribution;
- $t=t+1$; end.

Test:
input: trained models h_t and the corresponding β_t, $t=1,\ldots,H$;
For each test example, compute prediction $\tilde{y}_i$ by integrating outputs $\hat{y}_t$ of models h_t as

$$\tilde{y}_i = \begin{cases} \min\limits_{t=1,H}\left\{\hat{y}_t : \sum\limits_{j:\hat{y}_j \le \hat{y}_t} \log(1/\beta_j) \ge \frac{1}{2}\sum\limits_{j=1}^{H}\log(1/\beta_j)\right\} & \text{(weighted median) or} \\ \sum\limits_{t=1}^{H}\log(1/\beta_t)\hat{y}_t \,/ \sum\limits_{t=1}^{H}\log(1/\beta_t) & \text{(weighted mean) or} \\ \frac{1}{H}\sum\limits_{t=1}^{H}\hat{y}_t & \text{(simple mean) or} \end{cases}$$

Figure 1: AdaBoostC – a boosting-based algorithm for RBFN centers placement

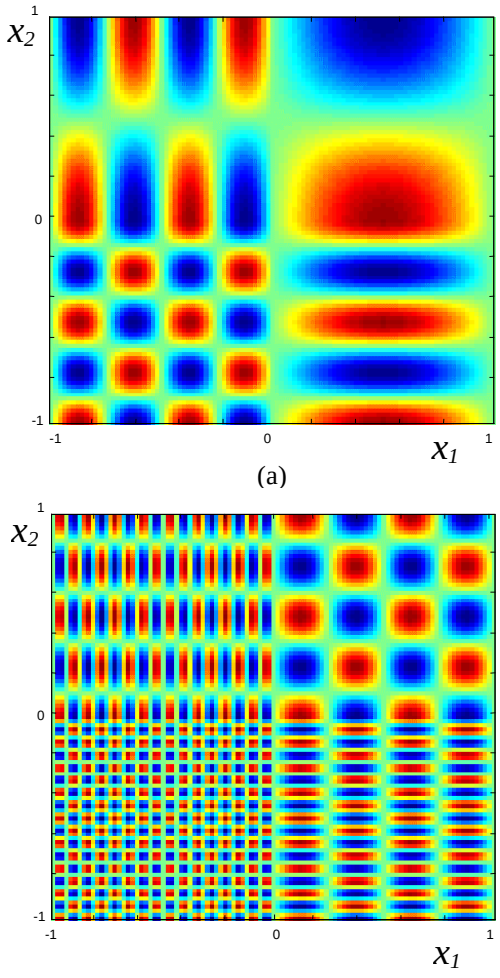


Figure 2: Functions approximated in the experiments with heterogeneous data

3. Results

For each experiment, the predicted function was specified by coefficients a_i,b_i of equation (1) . Each experiment was repeated 10 times using N_{train} training examples and 1000 test examples, randomly drawn according to a specified data distribution. For AdaBoostC, AdaBoostRT and AdaBoostSOM, boosting is performed with a pre-specified maximal number of T iterations. Prediction accuracy on test data is measured using the coefficient of determination R^2 computed as $R^2 = 1 - \frac{N-1}{N}\sum_i(\tilde{y}_i - y_i)^2 / \sum_i (y_i - \bar{y})^2$ [Devore, 1995], where $\tilde{y}_i$ and y_i are the i-th predicted and true target value, respectively, and $\bar{y}$ is the mean of all y_i's. R^2 is a measure of the explained variability of the target variable, where larger value is better with 1 corresponding to a perfect prediction and 0 to a trivial mean prediction. Finally, for each experiment, the sample mean and standard deviation of R^2 values on all test examples are reported as mean(R^2) ± std(R^2).

3.1 Learning heterogeneous function

The first experiment with a heterogeneous function was performed using equation (1) with $a_1=a_4=b_1=b_2=1$, $a_2=a_3=b_3=b_4=4$ to generate the data. This function is shown at Fig.2a. Both training and test data were uniformly distributed within the function domain.

Given a large training data set (N_{train} =600) the multilayer Perceptron with 50 hidden neurons, trained using the Levenberg-Marquart algorithm, had prediction accuracy R^2 = 0.91±0.05. Also, all RBFN algorithms worked well resulting in test R^2 values within (0.79, 0.92) range. This was much better than using a linear model [Devore, 1995], where the accuracy was comparable to a trivial mean predictor (R^2 = 0.00±0.02). However, when the training data size was decreased (N_{train}=100) the multilayer Perceptron, RBFN-SOM and RBFN-RT were also not able to provide prediction accuracy better than a trivial mean predictor. When RBFN-RT was applied, the constructed RBFNs had a small number of centers and so were unable to explain the test data. On the other hand, RBFN-SOM determined incorrect clusters due to a small number of examples and therefore its generalization was poor. Due to a small sample size, boosting these methods (AdaBoostRT and AdaBoostSOM) did not improve accuracy regardless of loss type, an output integration method and a drawing algorithm. For 100 training examples, RandomC and an optimized radius value r=0.3 was better than a trivial mean predictor when using less than 55 RBFN centers. However, AdaBoostC results with the same r were clearly superior for all number of RBFN centers (ranging from 5 to 100). The results shown in Fig.3 were obtained for linear loss, with the final model generated by a mean-based integration technique and using the roulette wheel method of random drawing. While AdaBoostC accuracy did not depend significantly on a drawing method, the results were influenced by a loss type and the output integration method, as shown in Table 1. It was interesting to observe that the weighted median method proposed in [Drucker, 1997] here provided the worst results. The choice of maximal number of iterations T for AdaBoostC was not critical, since on average the boosting process terminated due to achieving a loss value of 0.5 at about 9.5±1.2 iterations.

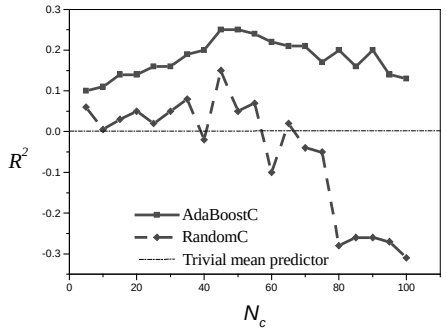


Figure 3: Prediction accuracy of AdaBoostC vs. RandomC trained on 100 examles of a heterogeneous function shown at Fig.2a

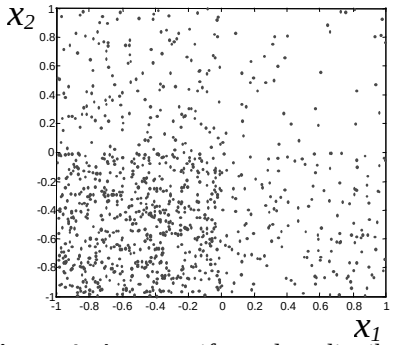


Figure 4: A non-uniform data distribution used for experiments reported in Section 3.2

Loss\Integration method	Weighted median	Mean	Weighted mean
Quadratic	0.14±0.07	0.15±0.09	0.21±0.09
Linear	0.17±0.13	0.25±0.05	0.20±0.05
Exponential	0.17± 0.06	0.21±0.06	0.22±0.10

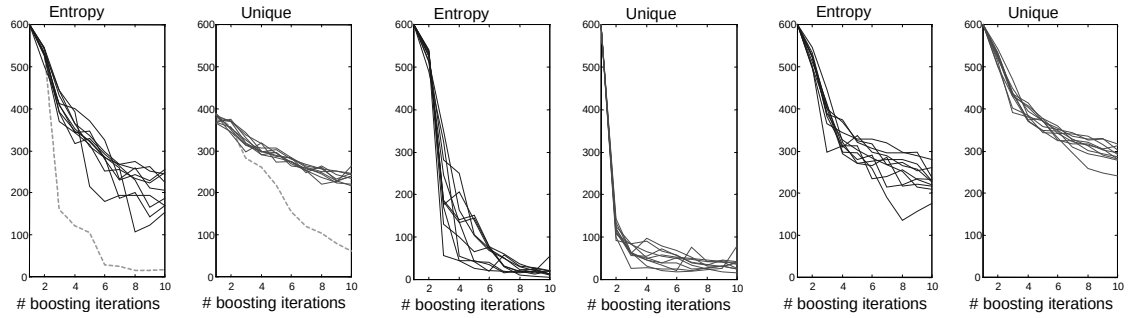
Table 1: Prediction accuracy (R^2) for AdaBoostC with N_{train}=100, T=10 boosting iterations, radius r=0.3 and N_c= 49 centers for a function shown at Fig. 2a

The previous experiment was repeated for a faster-changing function generated using equation (1) with $a_1=a_4=b_1=b_2=4$, $a_2=a_3=b_3=b_4=16$ (shown at Fig. 2b) where for N_{train}=600 similar results were obtained as for a small training set (N_{train}=100) in the previous experiment. AdaBoostC again achieved the best R^2 of 0.13±0.01, while RandomC gave R^2 of 0.04±0.02 and other considered algorithms, including the multilayer Perceptron and a linear model performed worse than a trivial mean predictor. Although a larger training set was used in this experiment, its size was still inadequate to explain the more complex function using the multilayer Perceptron and all considered RBFN methods except AdaBoostC.

3.2 Learning from a non-uniform data distribution

Next, experiments were performed on a non-uniform data distribution where the number of samples in the i-th quadrant was proportional to $a_i * b_i$, (see Fig.4). For a larger amount of training data (N_{train}=600) the multilayer Perceptron achieved prediction accuracy R^2 = 0.79±0.20 (the best results were obtained with 40 hidden neurons). Here, RBFN-RT and RBFN-SOM worked better than RandomC (with R^2 of 0.63±0.17, 0.53±0.07, and 0.42±0.07, respectively). In this case, AdaBoostRT provided a minor improvement of RBFN-RT (R^2 of 0.65±0.06), AdaBoostSOM provided a major improvement of RBFN-SOM when using deterministic sampling (R^2 0.68±0.08), while AdaBoostC improvement of RandomC was small but significant (R^2 of 0.50±0.04). On the other side, it is important to observe that using the roulette wheel and the stochastic remainder with replacement, the accuracy was consistently worse than when using a trivial mean predictor. To determine why this is the case, we traced an effective number of samples at the t-th boosting iteration measured by $2^{entropy(D_t)}$ (where $entropy(D_t)= \sum -D_t(i)\ log_2 D_t(i)$ is the distribution entropy at the t-th boosting iteration), and by the number of unique examples used for training at the t-th iteration. The results, shown in Fig.5, suggest that an accuracy drop can be explained by a drop in the effective number of examples since the boosting models trained on a small number of examples, although accurate on training, tend to have lower generalization abilities due to an overspecialization [Schapire at al, 1998].

For N_{train}=100, the obtained results were similar to these reported in the previous section for a small training set. With R^2 of 0.28±0.07 AdaBoostC outperformed other considered RBFN learning algorithms (R^2 of 0.17±0.10 was obtained for RandomC, -0.13±0.08 for RBFN-SOM, and -0.17±0.10 for RBFN-RT), a linear model (R^2 of -0.03±0.02) and the multilayer Perceptron (R^2 of -0.09±0.06).



a) roulette wheel b)stochastic remainder with replacement c)deterministic sampling

Figure 5: Influence of different sample drawing methods on an effective number of training samples during 10 repetitions of AdaBoostSOM

4. Conclusions

A new boosting-based method for RBFN centers placement is proposed. Its ability to improve prediction accuracy for regression of heterogeneous functions learned from sparse training sets is demonstrated by

comparisons to the alternative RBFN learning algorithms, the multilayer Perceptron, a linear model and a trivial mean predictor. Also, boosting directly applied to varying training data probability distributions in RBFN is shown to improve accuracy given a sufficiently large amount of data from a non-uniform distribution and using an appropriate boosting drawing algorithm.

This study has not attempted to determine potential advantages of various heuristics for identifying an appropriate number of radial basis centers and their parameters as these issues are addressed by our work in progress. We are also currently investigating an extension of the proposed RBFN method to classification.

References

Abu-Mostafa,Y., "Financial applications of learning from hints," *Advances in neural information processing systems*, Vol.7, 1995, pp. 411-418.

Carse,B.,T.Fogarty, "Fast evolutionary learning of minimal radial basis function neural networks using a genetic algorithm," *in Lecture notes in computer science*, Vol.1141, 1996, pp. 27-40.

Chen,S., "Nonlinear time series modeling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning," *Electronics letters*,Vol.31,No.2, 1995, pp. 117-118.

Cherkassky,V.,H.L.Najafi, "Constrained topological mapping for nonparametric regression analysis," *Neural networks*, Vol.4, No.1, 1991, pp. 27-40.

Devore,J.L., *Probability and statistics for engineering and sciences*, Duxbury Press, 1995.

Drucker,H., "Improving regressors using boosting techniques," *Proc. Fourteenth int'l conf. on machine learning*, 1997, pp. 107-115.

Drucker,H., "Boosting using neural nets," in *Combining artificial neural nets: ensemble and modular learning*, ed: A.J.C. Sharkey, Springer, 1999, pp. 51-77.

Freund,Y.,R.Schapire, "Experiments with a new boosting algorithm," *Proc. Thirteenth int'l conf. on machine learning*, 1996, pp. 148-156.

Friedman,J.H., "Multivariate adaptive regression spines," *The annals of statistics*, Vol.19, No.1, 1991, pp. 1-141.

Fritzke,B., "Fast learning with incremental radial basis function networks," *Neural processing letters*, Vol.1, No.1, 1994, pp. 2-5.

Goldberg,D.E., *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Reading, MA, 1989.

Hagan,M.,Menhaj,M.B.: "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. on neural networks*, Vol.5, 1994, pp. 989-993.

Haykin,S., *Neural networks, a comprehensive foundation*, Prentice-Hall 1999.

Kohonen,T., *Self-organization and associative memory*, Springer-Verlag, 1984.

Lowe,D., "Adaptive radial-basis function nanlinearities, and the problem of generalisation", *Proc. First IEE int'l. conf. artificial neural networks*, 1989, pp.171-175.

Orr,M.,J.Hllam,K.Takezawa,A.Murray,S.Ninomiya,M.Oide,T.Leonard, "Combining regression trees and radial basis function networks," *submitted to the International journal of neural systems*, 1999, http://www.anc.ed.ac.uk/~mjo/papers/rt99.ps.gz.

Pennington,M.,Volstad,J.H., "Optimum size of sampling unit for estimating the density of marine populations," *Biometrics*, Vol.47, 1991, pp. 717-723.

Poggio,T.,F.Girosi, "Networks for approximation and learning, " *Proc. IEEE*, Vol.78, No.9, 1990, pp. 1481-1497.

Saha,A.,J.Christian,D.S.Tang, "Oriented non-radial basis functions for image coding and analysis," *Advances in neural information processing systems*, Vol.3, 1991, pp. 728-734.

Schapire,R.E.,Y.Freund,P.Bartlett,W.S.Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The annals of statistics*, Vol.26, No.5, 1998, pp. 1651-1686.