Weighted Parzen Windows for Radial Basis Function Network Design

Gregory A. Babich and Leon H. Sibul

The Applied Research Laboratory, Penn State, PO Box 30, State College, PA 16804

Abstract

This paper shows how the weighted-Parzen-window (WPW) technique can be used for radial basis function network (RBFN) design. The WPW training algorithm uses an agglomerative hierarchical clustering procedure to find the RBFN centers and weights. This approach reduces storage requirements as it selects the centers and weights. It is shown that RBFNs can be designed using the WPW technique so that they are functionally equivalent to some statistical techniques. Experimental results are reported for two practical applications -- laser-weld classification and handwritten character recognition. The results show that WPW designed RBFNs outperform some neural techniques in these applications.

1: Introduction

Radial basis function networks (RBFNs) and related networks are often used to design pattern classification systems. The structure of the RBFN network is quite similar to that of the feed-forward artificial neural network (ANN) with a single hidden layer. Therefore, RBFNs have been explored as alternatives to traditional ANNs. Furthermore, the well-known limitations of the error-back-propagation (EBP) [10] technique have also encouraged the use of RBFNs in neural-type applications. Commonly cited EBP shortcomings include susceptibility to local minima, failure to converge, and long training times [2], [7], [8], [11], [12]. Some RBFN shortcomings have also been noted. Two challenging issues are excessive storage requirements and selection of the RBF centers [2], [8], [11], [12].

This paper shows how the WPW technique can be used for RBFN classifier design. The WPW training algorithm uses an agglomerative (bottom-up) hierarchical clustering procedure to find the RBFN centers and weights. In general, there are fewer centers than training samples because of the agglomerative clustering. The WPW approach thus helps to overcome the main limitations of RBFNs which are excessive storage and difficulty in selecting the RBF centers. Furthermore, it is shown that the WPW algorithm can easily lead to RBFNs that are functionally equivalent to several statistical classifiers with well-known properties. These results clearly establish the link between RBFN and statistical techniques. Experimental results are presented for laser-weld classification and handwritten character recognition applications. The WPW is compared with the Parzen-window, error-back-propagation, and probabilistic-neural-network classifiers.

Section 2 briefly reviews the RBFN. Section 3 shows how a RBFN can be designed using the WPW technique. Section 4 demonstrates the relation between the WPW approach and several statistical techniques. Experimental results are presented in Section 5.

2: Radial basis function networks

The radial basis function network (RBFN) response is characterized by the function

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \varphi\left(\left\|\mathbf{x} - \mathbf{c}_i\right\|\right), \qquad (2.1)$$

where $\mathbf{x}$ is a d-dimensional input, $\varphi(\cdot)$ is a RBF, $\|\cdot\|$ is the Euclidean norm, $\mathbf{c}_i$ is the ith d-dimensional RBF center, and λ_i the corresponding weight. The Gaussian RBF is often used in (2.1) and similar networks [6], [7], [8], [9], [11], [12]. The radially-symmetric Gaussian density function is given by

1058-6393/95 $4.00 © 1995 IEEE

$$\varphi\left(\|\mathbf{x}-\mathbf{c}_i\|\right)=\frac{1}{(2\pi)^{\frac{d}{2}}h^d}\exp\left(-\frac{1}{2}\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{h^2}\right), \quad (2.2)$$

where h is the width parameter. The width parameter can be used to vary (2.2) from a spike for extremely small values of h to a hyperplane for extremely large values. While a more general form of (2.2) exists, it will not be explored in this paper. Various techniques have been proposed to select $\mathbf{c}_i$ and λ_i. These techniques include clustering, least-mean-square, orthogonal least-square, pseudo-inverse, and error-back-propagation procedures [2], [6], [7], [8].

3: Weighted Parzen windows

The weighted-Parzen-window (WPW) approach is directly related to the classic work of Parzen [9]. Therefore, the Parzen-window (PW) density estimate is briefly reviewed. Then, the WPW training and classification procedures are presented. For notational simplicity, the procedures discussed below address only a single class of data. The multiclass generalization is given when the classification procedure is discussed.

Given a set of n d-dimensional training vectors $X=\{\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_n\}$, the Parzen-window probability density function estimate is given by

$$f_n(\mathbf{x})=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}\kappa\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right), \quad (3.1)$$

where $\kappa(\cdot)$ is the *window* function and h is the width parameter. Equation (3.1) converges to the true density if h and $\kappa(\cdot)$ are properly selected, and the distribution of $\mathbf{x}$ is continuous [3], [4], [9]. The width parameter is required to be a function of n such that

$$\lim_{n\to\infty} h^d(n)=0$$

and

$$\lim_{n\to\infty} nh^d(n)=\infty.$$

The window function is required to be a finite-valued non-negative density function [4], [5], where

$$\int \kappa(\mathbf{y})d\mathbf{y}=1.$$

Many window functions are possible. Rectangular and Gaussian window functions are commonly used [3], [4], [5], [9]. The Gaussian window function is given by

$$\kappa(\mathbf{y})=\frac{1}{(2\pi)^{\frac{d}{2}}}\exp\left(-\frac{1}{2}\|\mathbf{y}\|^2\right). \quad (3.2)$$

The WPW method is used to approximate (3.1) with fewer centers. The WPW approximation is given by

$$f_m(\mathbf{x})=\frac{1}{n}\sum_{i=1}^{m}\frac{w_i}{h^d}\kappa\left(\frac{\mathbf{x}-\mathbf{c}_i}{h}\right), \quad (3.3)$$

where w_i is the ith window weight, $\mathbf{c}_i$ is the ith center, and $m \le n$. When (3.2) is used (3.3) becomes

$$f_m(\mathbf{x})=\frac{1}{n}\sum_{i=1}^{m}\frac{w_i}{(2\pi)^{\frac{d}{2}}h^d}\exp\left(-\frac{1}{2}\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{h^2}\right)$$
$$=\sum_{i=1}^{m}\lambda_i\varphi\left(\|\mathbf{x}-\mathbf{c}_i\|\right), \quad (3.4)$$

where $\lambda_i=n^{-1}w_i$. Note that (3.4) is in the form of (2.1) which clearly indicates the *structural* similarity between the WPW and radial basis function networks. The WPW training algorithm, given below, is used to find the centers and weights $C=\{\mathbf{c}_1\ \mathbf{c}_2\ \ldots\ \mathbf{c}_m\}$ and $\mathbf{w}=[w_1\ w_2\ \ldots\ w_m]^T$.

1. Choose: h, $0.0 \le e_{max}$.
2. Initialize: $m \leftarrow n$, $C \leftarrow X$, $w_i=1$, $i=1,\ldots,m$.
3. Choose the closest centers $\|\mathbf{c}_j-\mathbf{c}_k\|$, $j \ne k$.
4. Calculate: $\mathbf{c}_o=\dfrac{w_j\mathbf{c}_j+w_k\mathbf{c}_k}{w_j+w_k}$.
5. Update: $\{\mathbf{c}_j, \mathbf{c}_k\} \notin C$, $\mathbf{c}_o \in C$, $w_o \leftarrow w_j+w_k$, $m \leftarrow m-1$.
6. Calculate: $e=\dfrac{1}{n}\sum_{i=1}^{n}\dfrac{|f_n(\mathbf{x}_i)-f_m(\mathbf{x}_i)|}{f_n(\mathbf{x}_i)}$.
7. IF [($e < e_{max}$) AND ($m > 1$)] go to 3.
8. IF ($e > e_{max}$) Replace: $\mathbf{c}_j$, $\mathbf{c}_k$, w_j, w_k.
9. Output: C and $\mathbf{w}$.

As can be seen by the algorithm, the WPW training is an agglomerative hierarchical clustering procedure [3] where the centers are the mean of each cluster. Note that the number of centers are determined automatically based on the parameters h and e_{max}. Furthermore, since agglomerative clustering is used, the number of centers necessarily decreases with each training step. Therefore, the

WPW procedure reduces storage requirements as it selects the centers. A clear advantage of the WPW training algorithm is that it finds the centers and weights simultaneously unlike other cluster-based approaches which find the centers by clustering and the weights by some other means (see references [7] and [8] , for example). Selection of h and e_{max} are discussed in Section 5.

After training, classification can begin. Given that there are c classes of data labeled $\omega_1, \omega_2, \ldots, \omega_c$, unlabeled samples $\mathbf{x}$ can be given the class label ω_i according to the largest valued *discriminant* function

$$g_i(\mathbf{x}) = f_m(\mathbf{x}|\omega_i)P(\omega_i),$$

where $f_m(\mathbf{x}|\omega_i)$ is the class-conditional WPW density approximation (3.4) and $P(\omega_i)$ is the ith class's a priori probability. This procedure is a Bayes optimal approach [3]. However, optimality is only guaranteed in certain cases which are discussed in the following section.

4: Relation to statistical classifiers

Statistical classifiers can be designed by the weighted-Parzen-window (WPW) algorithm in certain cases. This only requires that the width parameter, h, error parameter, e_{max}, radial basis function, and a priori probabilities, $P(\omega_i)$, $i = 1, \ldots, c$, are selected for the desired results. Full derivations are omitted for brevity, but they are straight-forward (see [1]). By using certain parameters and the Gaussian density function, the following specific classification performance can be achieved as special cases of the WPW algorithm: Parzen-windows (PW), 1-nearest-neighbor (NN), Bayes-Gaussian, and minimum-distance (MD) [3], [4], [5] [9]. Table 1 shows the parameter choices necessary for each of the classifiers. Note that the same value of e_{max} is used for each class.

Table 1: Statistical classifier design using WPWs.

h_i	e_{max}	$P(\omega_i)$	Classifier
h_i	0	$P(\omega_i)$	PW
$h = 0^+$	0	$P(\omega) = 1/c$	NN
h_i	> 2	$P(\omega_i)$	Bayes-Gaussian
$h_i = h$	> 2	$P(\omega) = 1/c$	MD

Designing the Parzen-window classifier using the WPW procedure represents the extreme case of tolerating no error as measured by Step 6 of the training algorithm. In this case, the WPW classifier is exactly the PW classifier which is Bayes optimal in the large sample case. The NN classifier is simply a special case of the PW classifier. This result is mainly due to the very small value of h which causes the classifier to behave like the NN classifier [11]. The NN classifier has an error rate which is less than twice the Bayes rate in the large sample case [3].

The other extreme case is when all error (Step 6) is tolerated. In this case, the WPW training algorithm terminates with a single vector for each class which are the sample means. So, a single Gaussian RBF will be centered at the mean of each class. In this case, the WPW procedure can be used to design the Gaussian classifier. The Gaussian classifier, in this form, is Bayes-optimal for Gaussian data with covariance matrices given by $\Sigma_i = h_i^2 I$, where I is the identity matrix and $i = 1, \ldots, c$. The minimum-distance classifier is a special case of the Gaussian classifier and is only Bayes-optimal if all covariance matrices are proportional to I and equal, i.e., $\Sigma_i = \Sigma = h^2 I$ [3].

The results given by Table 1 represent limiting cases, i.e., when one center or all of the training vectors remain for each class. They clearly show the relationship between the WPW designed RBFN and well-known statistical classifiers. However, these results are primarily of theoretical importance since in practical applications, we are often interested in reducing the number of centers but not necessarily to a single vector. Experimental results are necessary to demonstrate the WPW approach in the "in-between" cases.

5: Experimental results

Experiments were conducted with the weighted-Parzen-windows (WPW) approach for two real data sets. Also, comparative experiments were conducted using the Parzen-window (PW) [9], error-back-propagation (EBP) [10], and probabilistic-neural-network (PNN) [11] classifiers. In each case, system parameters were selected, then the error rate was found using the leave-one-out (LV) [3], [4] technique. For

the WPW, PW, and PNN classifiers, Gaussian density functions (3.2) were used and the a priori probabilities were set equal for each class.

The first data set used was extracted from the acoustic signature of a laser-welder. A single feature, energy, is used for two classes of weld which are full- and partial- penetration. This data set has 525 full-penetration samples and 482 partial-penetration samples. The second data set consists of 12-dimensional feature vectors which were extracted from handwritten characters. This data set has three classes which are the uppercase letters A, R, and L. There are 48 feature vectors for each class.

The WPW classifier was trained by first finding the width parameter h, and then the value of e_{max}. In both experiments, h and e_{max} were the same for each class; however, separate values can be used. Selection of the width parameter, h, was accomplished by varying it over several orders of magnitude, finding the LV error rate for the PW classifier, and choosing the value h_{opt} corresponding to the minimum error [5]. Several values of e_{max} were selected. To aid in the selection of e_{max}, the density error (Step 6) was plotted as a function of the training level. The plot was then used to select values of e_{max} which trade-off density error for storage savings.

The PW classifier does not involve training. It only requires selection of h. The PNN is a special case of the PW classifier. Training of this classifier is accomplished by normalizing the training data to unit length [11], [12]. A width parameter is also necessary for the PNN. The value of h_{opt} was selected for each classifier as discussed above. Both of these classifiers require storage of the entire training set.

The EBP structure used in this experiment had an input layer, hidden layer, and output layer [10]. The training parameters were the learning constant and momentum factor. The steepness factor of the sigmoid activation function was taken to be 1. The classifier's parameters were selected by trial and error to minimize the resubstitution [4] error rate. The laser-weld classifier used 11 neurons in the hidden layer while the handwritten character classifier used 46 neurons in the hidden layer.

Table 2 shows that in all cases, the WPW error rate is the same or lower than any of the other classifiers. Also, it was noted that significant storage savings could be gained without increasing the error rate of the WPW classifier. In the case of the handwritten characters, the WPW storage requirements were less than 50% of the PW and PNN classifiers. In the case of the laser-weld data, the WPW storage requirements were less than 2.0% of the PW classifier. Note that the PNN cannot be used in this case of the laser-weld data because normalization of the 1-dimensional data maps them into a single point -- +1. Also, it was noted, for the values of e_{max} shown in Table 2, that the WPW classifier storage requirements were comparable to those of the EBP, but the error rates are smaller for the WPW.

Table 2: Experimental results.

Data	Laser-weld		ARL	
Classifier	e_{max}	LV (%)	e_{max}	LV (%)
WPW	0.05	7.5	0.10	1.4
	0.10	7.5	0.20	1.4
	0.15	7.4	0.30	1.4
PW	--	7.5	--	1.4
PNN	--	NA	--	1.4
EBP	--	18.1	--	5.6

6: Conclusion

This paper shows how the weighted-Parzen-window (WPW) technique can be used for radial basis function network (RBFN) classifier design. The WPW training algorithm uses a hierarchical clustering procedure to find the RBFN centers and weights. This approach reduces storage requirements as it selects the centers and weights. It was shown that the WPW training algorithm can be used to design several well-known statistical classifiers as special cases of the WPW classifier. Experimental results showed that the WPW classifier outperformed some artificial neural networks in two practical applications. The error rate of the WPW was the same or less than all other classifiers used. Also, it was noted that the storage requirements can be significantly less than other classifiers.

Acknowledgments

We thank the Applied Research Laboratory at Penn State, who, under contract of the Office of Naval Research, partially funded this work. Furthermore, we gratefully acknowledge the comments of Professors N. K. Bose, O. I. Camps, and R. L. Tutwiler. In addition, we are very grateful to D. F. Farson and K. L. Hillsley who provided the laser-weld data. Thanks also to A. G. Pisani for proofreading the manuscript.

References

[1] G. A. Babich, "Weighted Parzen Windows for Pattern Classification," M.S. Thesis, Penn State University, May 1994.

[2] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, Vol. 2, No. 2, pp. 302-309, March 1991.

[3] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, NY, 1973.

[4] K. Fukunaga, *Statistical Pattern Recognition*, 2d ed., San Diego, Academic Press Inc., 1990.

[5] A. K. Jain and M. D. Ramaswami, "Classifier Design with Parzen Windows," in *Pattern Recognition and Artificial Intelligence*, pp. 211-228, E. S. Gelsema and L. N. Kanal, eds. Elsevier Science Publishers B.V. (North-Holland), 1988.

[6] S. Lee and R. M. Kil, "A Gaussian Function Network with Hierarchically Self-Organizing Learning" *Neural Networks*, Vol. 4, pp. 207-224, 1991.

[7] J. Moody and C. Darken, "Fast-Learning in Networks of Locally Tuned Processing Units," *Neural Comput.*, Vol. 1, No. 2, pp. 281-294, 1989.

[8] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the Training of Radial Basis Function Classifiers," *Neural Networks*, Vol. 5, pp. 595-603, 1992.

[9] E. Parzen, "On Estimation of a Probability Density Function and Mode," *Ann. Math. Stat.*, Vol. 33, pp. 1065-1076, September 1962.

[10] D. E. Rumelhart, J. L. McClelland and the PDP Research Group, *Parallel Distributed Processing, Vol. 1, Foundations*, M.I.T. Press, 1986.

[11] D. F. Specht, "Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification," *IEEE Transactions On Neural Networks*, Vol. 1, No. 1, pp. 111-121, March 1990.

[12] D. F. Specht, "Applications of Probabilistic Neural Networks," *SPIE Vol. 1294 Applications of Artificial Neural Networks*, pp. 344-353, 1990.