Gap-Based Estimation: Choosing the Smoothing Parameters for Probabilistic and General Regression Neural Networks

M. Zhong, D. Goggeshall, E. Ghaneie, T. Pope, M. Rivera, M. Georgiopoulos, G. Anagnostopoulos, M. Mollaghasemi, and S. Richie

Abstract—**Probabilistic Neural Networks (PNN) and General Regression Neural Networks (GRNN), both proposed by Specht, are well known architectures for classification and regression problems, respectively. They represent the knowledge by a simple but interpretable model, which approximates the optimal classifier / predictor in the sense of misclassification rate / mean square error, assuming random inputs. Both models require a preset parameter, called smoothing parameter, which can be uniform or variable among dimensions, instances, and/or classes. Previous research has shown that this parameter is, to some extent, important to the accuracy of both PNN and GRNN. Usually the smoothing parameter is chosen by cross-validation or clustering. In this paper, however, we demonstrate the difficulties of both these approaches, discuss the relationship between this parameter and some of the data statistics, and attempt to develop a fast approach to determine the optimal value of this parameter. We explain why this parameter can vary in a certain range, while maintaining the network's accuracy, as most experiments have shown, and even provide the quantitative expression of this range. Finally, through experimentation we show that our approach, referred to as a gap-based estimation approach, is superior to the cross validation approach.**

Manuscript received December 9, 2005. This work was supported in part by the National Science Foundation (NSF) under grants CRCD 0203446 and CCLI 0341601.

M. Zhong is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: myzhong@ucf.edu).

D. Goggeshall was with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: david.coggeshall@gmail.com).

E. Ghaneie was with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: Ehsan.Ghaneie@gmail.com).

T. Pope was with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: ThomasPope@gmail.com).

M. Rivera was with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: mark.rivera@gmail.com).

M. Georgiopoulos is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (phone: (407) 823-5338, fax: (407) 823 5835; e-mail: michaelg@mail.ucf.edu).

G. Anagnostopoulos is with the Department of Electrical and Computer Engineering, Florida Institute of Technology, Melbourne, FL 32901, USA (e-mail: georgio@fit.edu).

M. Mollaghasemi is with the Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, FL 32816, USA (e-mail: mollagha@mail.ucf.edu).

S. Richie is with the School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA (e-mail: richie@mail.ucf.edu).

I. Introduction

CLASSIFICATION and regression are two types of common problems in science, technology, and even our daily life. Assuming that the inputs/outputs have a fixed and known relationship expressed by the conditional probabilities, it can be proven that the optimal classifier is the Bayes classifier and the optimal predictor is expressed by the conditional expected value of the output given the inputs. In practice, the probabilities representing the input/output relationship are not known but are estimated from the given training instances, by using legitimate approaches for their estimation. One such probability estimation approach is the Parzen window approach [1], extended by Cacoullos [2], which estimates the class conditional probabilities as sum of Gaussian functions centered at the training points with appropriately chosen widths (variances), designated from now on as smoothing parameters. Parzen's approach works well under some reasonable assumptions regarding the choice of the smoothing parameters and when the training data becomes very large the approximation becomes equal to the actual class conditional probabilities. In the case though where the training data sets are of finite size (as it is usually the case in practice) the right choice of the smoothing parameters is essential for the good performance of the classifier. PNN, invented by Specht [3], approximates Bayes classifier where the class conditional probabilities are estimated by using the Parzen's approach. GRNN, also invented by Specht, is the PNN's regression counterpart [4].

One of the major issues associated with the PNN/GRNN is how to choose the smoothing parameter(s) involved in the Gaussian functions utilized to estimate the conditional probabilities. Specht suggested using cross validation to estimate the smoothing parameters in [5] and [6]. This approach, although simple, has two major difficulties:

1) Choosing the right candidate values for validation is not as simple as one expects. Although the smoothing parameter represents the standard deviation in the Gaussian kernels, we show that in a one-dimensional example (i.e., each input has only one attribute), the smoothing parameter should be chosen less than the standard deviation of the observed instances. In fact, the optimal smoothing parameter can be much smaller than the standard deviation of the observed instances. Consequently, the theoretical range of the optimal smoothing parameter is infinitely wide if one chooses the smoothing parameter based on the standard deviation only.

2) Cross-validation is very time-consuming. The time complexity of cross-validation in PNN/GRNN is $O(D \cdot PT \cdot PV)$, where D is the dimensionality of the input, PT is the number of instances in the training set, and PV is the number of instances in the validation set. If N candidate smoothing parameters are examined, the time complexity of cross-validation is $O(N \cdot D \cdot PT \cdot PV)$.

Another way of dealing with this issue of smoothing parameters is to cluster the training data and approximate the class conditional probabilities by Gaussian functions centered at the cluster points instead of the actual training points. The clustering of the data gives the additional capability of estimating the smoothing parameters of these Gaussian functions as the within-cluster standard deviations. Clustering procedures that have been used in the literature in relation to the PNN neural network are: LVQ approach [7], K-Means clustering [8], and mixture of Gaussians [9]. In this paper, we apply an unsupervised variant of Gaussian ARTMAP (GAM) [10] [11] to cluster the data. We call this clustering variant Gaussian ART (GART). Although this approach compresses the training data and may determine a better value for the smoothing parameter than cross-validation, it tends to deteriorate the estimate of the probability density functions (PDFs), as shown in this paper.

Our approach attempts to determine a good estimate of the optimal smoothing parameters with a time complexity of $O(D \cdot PT)$. No cross-validation is required. This is a significant computational advantage and it also an advantage in practical situations, where the dataset given is small, and we do not have the luxury of defining a sizable cross-validation set.

The organization of the paper is as follows: Chapter II introduces PNN and GRNN, including their clustered versions; Chapter III discusses the effect of the smoothing parameter on PNN and GRNN; Chapter IV describes our approach of choosing the smoothing parameter; Chapter V compares our approach of choosing the smoothing parameter to other commonly used approaches; Chapter VI provides a summary of our work and conclusive remarks. For the rest of the paper we assume that the reader is familiar with GAM.

II. Preliminaries

A. Probability Neural Network (PNN)

The Bayes classifier is based on the following formula for finding the class that a datum $\mathbf{x}$ belongs.

$$P(c_j \mid \mathbf{x}) = \frac{f(\mathbf{x} \mid c_j)P(c_j)}{f(\mathbf{x})} \tag{1}$$

The above probabilities for every class j in our pattern classification task is calculated, and the datum $\mathbf{x}$ is classified as belonging to the class j that maximizes this probability. In order to calculate the above probabilities one needs to estimate the class conditional probabilities $f(\mathbf{x}|c_j)$ and the a-priori probabilities $P(c_j)$ for every class j (the calculation of $f(\mathbf{x})$ is not needed because it is a common factor in all of these probabilities and can be cancelled out). The a-priori probabilities $P(c_j)$ are calculated from the given training data. The class conditional probabilities $f(\mathbf{x}|c_j)$ can also be calculated from the training data by using the approximation suggested in [1]. In particular, the following approximation is utilized to estimate these class conditional probabilities in [3]:

$$f(\mathbf{x} \mid c_j) = \frac{1}{(2\pi)^{D/2}\sigma^D PT_j}\sum_{r=1}^{PT_j}\exp\left[-\frac{(\mathbf{x}-\mathbf{X}_r^j)^T(\mathbf{x}-\mathbf{X}_r^j)}{2\sigma^2}\right] \tag{2}$$

where D is the dimensionality (number of attributes) of the input patterns, PT_j represents the number of training patterns belonging to class j, $\mathbf{X}_r^j$ denotes the r^{th} such training pattern, $\mathbf{x}$ is the input pattern to be classified, and σ is the smoothing parameter that we talked about earlier. It is pointed out in [2], that this estimation will approach asymptotically the real PDF if both the following equations are true:

$$\lim_{PT_j \to \infty} \sigma = 0 \tag{3}$$

$$\lim_{PT_j \to \infty} PT_j \sigma = \infty \tag{4}$$

Equation (2) is the basis of the PNN classifier. The smoothing parameter σ should be selected properly, as further discussed in Chapter III. In general, the smoothing parameters may depend on the dimension and the class of the data. For simplicity, we assume independence among attributes, so that the smoothing parameter does not become a matrix for each class. The above formula for the estimation of the class conditional probabilities now becomes.

$$f(\mathbf{x} \mid c_j) = \frac{1}{(2\pi)^{D/2}\prod_{i=1}^{D}\sigma_{ij} PT_j}\sum_{r=1}^{PT_j}\exp\left[-\sum_{i=1}^{D}\frac{\left(x_i - X_{ir}^j\right)^2}{2\sigma_{ij}^2}\right] \tag{5}$$

where σ_{ij} is the smoothing parameter across dimension i for the training points belonging to class j.

B. General Regression Neural Network (GRNN)

For regression problems, it is not difficult to prove that the solution minimizing the expected mean square error is the conditional expected value given below:

$$E(y \mid \mathbf{x}) = \frac{\int_{-\infty}^{\infty} y\, f(\mathbf{x}, y)dy}{\int_{-\infty}^{\infty} f(\mathbf{x}, y)dy} \tag{6}$$

In practice, the joint PDF $f(\mathbf{x},y)$ is estimated by:

$$f(\mathbf{x}, y) = \frac{\sum_{r=1}^{PT}\exp\left[-\frac{\left(y-y_r\right)^2}{2\sigma_0^2} - \sum_{i=1}^{D}\frac{\left(x_i - X_{ir}\right)^2}{2\sigma_i^2}\right]}{PT\left(2\pi\right)^{(D+1)/2}\prod_{i=0}^{D}\sigma_i} \tag{7}$$

Equations (6) and (7) indicate that $E(y|\mathbf{x})$ can be estimated by:

$$E(y \mid \mathbf{x}) = \frac{\sum_{r=1}^{PT} y_r \exp\left[-\sum_{i=1}^{D}\frac{\left(x_i - X_{ir}\right)^2}{2\sigma_i^2}\right]}{\sum_{r=1}^{PT}\exp\left[-\sum_{i=1}^{D}\frac{\left(x_i - X_{ir}\right)^2}{2\sigma_i^2}\right]} \tag{8}$$

where σ_i is the smoothing parameter across dimension i.

C. *Clustered PNN and GRNN*

As shown above, both PNN and GRNN simply memorize all the training instances to perform classification or regression. To compress the training instances, one can cluster them and represent them by the clusters centers. Thus, each cluster center is essentially a training instance with a certain multiplicity. When applied to PNN/GRNN, we should expect the smoothing parameter to depend on the clusters. It is not difficult to modify (5) for PNN to the following:

$$f(\mathbf{x}\,|\,c_j) = \frac{\displaystyle\sum_{r=1}^{N^j} \frac{N_r^j}{\prod_{i=1}^{D} \sigma_{ir}^j} \exp\left[-\sum_{i=1}^{D} \frac{\left(x_i - \overline{X}_{ir}^{\,j}\right)^2}{2\sigma_{ir}^{j\,2}}\right]}{\displaystyle (2\pi)^{D/2} \sum_{r=1}^{N_j} N_{jr}} \tag{9}$$

where N_r^j is the multiplicity of, or the number of original training instances covered by, the rth cluster in class *j*, while σ_{ir}^j and $\overline{X}_{ir}^{\,j}$ are the smoothing parameter and the mean value of the ith dimension for the rth cluster in class *j*, respectively.

Similarly, Clustered GRNN can be described by

$$E(y\,|\,\mathbf{x}) = \frac{\displaystyle\sum_{r=1}^{PT} \frac{y_r N_r}{\prod_{i=1}^{D} \sigma_{ir}} \exp\left[-\sum_{i=1}^{D} \frac{\left(x_i - \overline{X}_{ir}\right)^2}{2\sigma_{ir}^{\,2}}\right]}{\displaystyle\sum_{r=1}^{PT} \frac{N_r}{\prod_{i=1}^{D} \sigma_{ir}} \exp\left[-\sum_{i=1}^{D} \frac{\left(x_i - \overline{X}_{ir}\right)^2}{2\sigma_{ir}^{\,2}}\right]} \tag{10}$$

III. Analysis of Smoothing Parameter In One-dimensional Cases

A. *Non-Clustered Cases*

For simplicity, let us first consider only one dimension and one class without clustering. In this case, (5) reduces to

$$f(x) = \frac{1}{(2\pi)^{1/2}\sigma PT} \sum_{r=1}^{PT} \exp\left[-\frac{\left(x - X_r\right)^2}{2\sigma^2}\right] \tag{11}$$

It is not difficult to prove that:

$$VAR\left(\hat{X}\right) = VAR\left(X\right) + \sigma^2 \tag{12}$$

where the left hand side represents the expected value of the variance of the point *x* using the estimated PDF given in (11), and *VAR*(*X*) stands for the expected value of the variance of the point *x* using the true PDF. In order to produce an accurate estimate of the PDF, a necessary condition is:

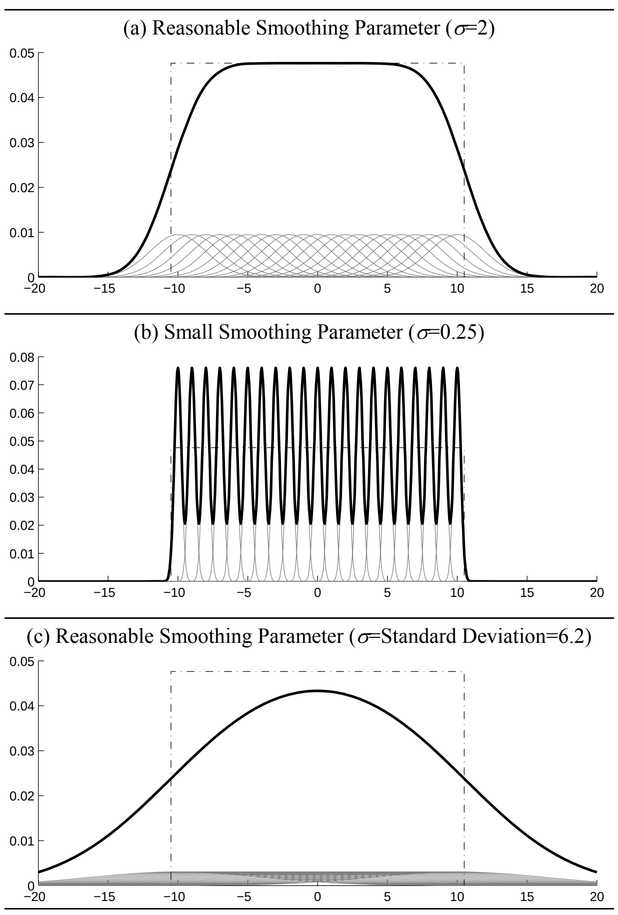
$$\sigma << STD(X) \tag{13}$$

where *STD*(*X*) is the unbiased standard deviation of *X*. This conclusion agrees with (3), since the standard deviation usually approaches a positive constant number when the number of instances goes to infinity.

Note that setting σ_i to a large value can smooth out the ith attribute, making the corresponding marginal PDF constant (almost zero) in the whole space. We do not, however, consider this method beneficial, since it alters the PDF too much and it tends to scale the final PDF to almost zero for all classes if an attribute is not relevant to any class. Thus, when an attribute is completely noise, we still estimate the PDF as distributed in the range shown in the training set, which is large enough to smooth out this attribute given that the training set is representative.

On the other hand, σ_i cannot be too small, or otherwise the PDF becomes spiky (consists of a number of impulse functions) at the training points and is almost zero at other places, which causes the over training problem.

To understand the effect of the smoothing parameter better, consider an example where the training set is {-10, -9, …, 10}, and is taken from a uniform distribution in the range [-10.5, 10.5]. (In practice, even if the points are uniformly distributed, the observed instances are not likely to be equally spaced; here we simply use this ideal case to demonstrate our idea and we will discuss the practical case later.) Fig. 1 shows the resulting estimated PDF in three typical cases. It verifies our previous statements. It also shows that the standard deviation is usually too large to be used as the smoothing parameter. In fact, this problem becomes even worse as the distribution is fixed and the number of instances is increased.

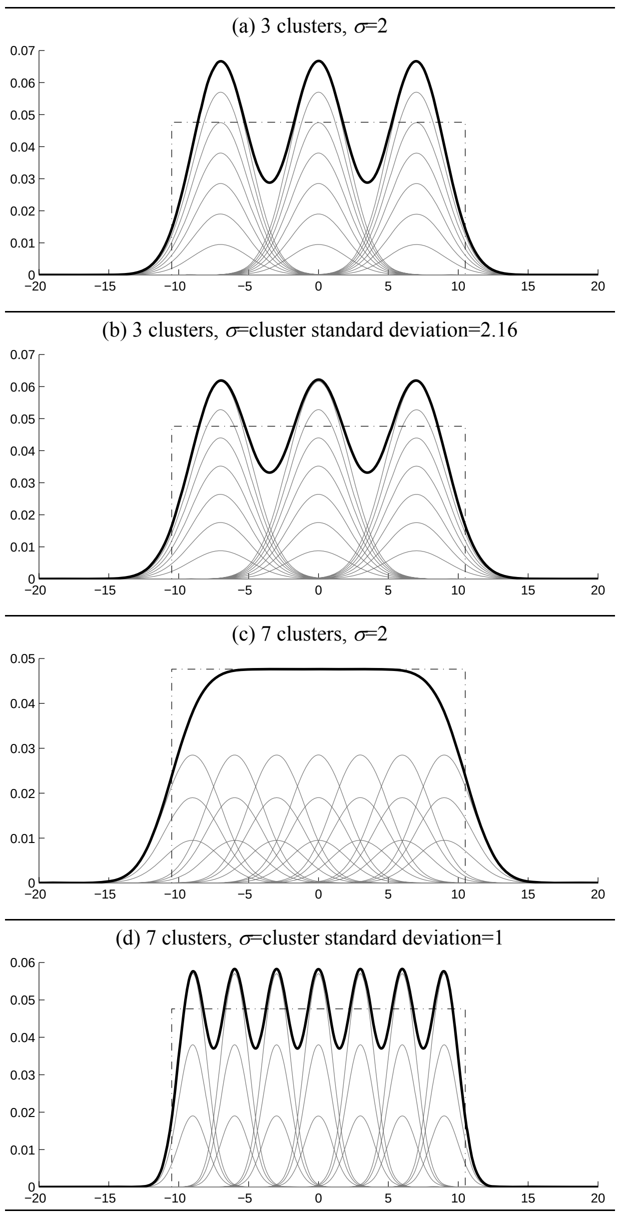


Dash-dot line: desired (true) PDF
Gray solid line: Gaussian function centered at a training point
Black solid line: estimated PDF

Fig. 1. Estimates of PDF with various σ value (no clustering)

B. *Clustered Case*

Suppose the previous training set with 21 instances is clustered, with each cluster containing the same number of instances. Fig. 2 shows the effect of the cluster size and the smoothing parameter on the PDF estimation.



Dash-dot line: desired (true) PDF
Gray solid line: Gaussian function centered at a training point (for better visualization, the Gaussian functions in each cluster are summed up instead of being overlapped)
Black solid line: estimated PDF

Fig. 2. Estimates of PDF with various σ value (no clustering)

As shown above, neither the cluster standard deviation nor reasonable smoothing parameter in the previous example yields an accurate PDF for both the 3-clustered and the 7-clustered case. Since clustering represents multiple instances by their mean value, and as a result loses considerable information, it is difficult to determine a good smoothing parameter quickly.

IV. Our Approach: Gap-Based Estimation

A. *Result from a Simple Model*

Fig. 1 indicates that the width of Gaussian kernels, which is directly controlled by the smoothing parameter, should be related to the gap between two nearest points. Our proposed approach, the Gap-based estimation, is originated from the following simple idea: when the training points are equally spaced in (-∞, 0], the estimated PDF should:

1) Be almost constant on the left of the origin, and

2) Drop to almost zero at the points on the right of the origin, although we allow a transition interval where the value drops from the constant value to the zero value.

Numerical solution of the above two constraints indicates that $0.69d<\sigma<2.18d$, where d is the distance between two neighbors in the above model. In practice, the training points are seldom equally spaced even if uniform distributed, and thus we double the bounds as $1.38d<\sigma<4.36d$. We usually choose $\sigma=4d$ for better generalization.

B. *Sampling in Multi-Dimensional Space*

For multi-dimensional cases, we should standardize each dimension first so that each dimension has unity standard deviation before computing the gap, which prevents bias towards large-scale dimensions. In these cases,

$$\sigma_i = \min(4d,\, 0.5)STD(X_i) \qquad (14)$$

where d is the average gap (roughly the average value of the minimum distance) between two input points after standardization and $STD(X_i)$ is the standard deviation of the ith dimension before standardization. Note that $4d$ might be larger than 1 when the dimensions are not independent, which is a challenge to us since the assumption of (5) is violated. In this case, we replace $4d$ to 0.5 to retain the validity of (13).

Computing d directly has a high computational complexity of $O(PT^2)$. Therefore, we estimate d by sampling the points. We randomly choose M points, where $M << PT$ when PT is large, and for each one (**X**) of these points we calculate the largest value of the minimum $2D$ distances to another small sample of N points, where $N << PT$ when PT is large. However, two issues are raised due to sampling.

First, if the points are distributed in clusters, it is possible that all the N points are in a different cluster than the **X**, which causes d to be estimated as the distance between clusters, despite the fact that we desire to obtain the local distance among the points in the same cluster. Nevertheless, we assume each cluster has the same number of points, which is at least $4C$ where C is the number of clusters. Hence,

$$PT \ge 4C^2 \qquad (15)$$

In this case, if we set N to $4\sqrt{PT}$, we can prove the probability that none of the N sampled points are in the same cluster as **X** does not exceed $e^{-8}=3.35\times10^{-4}$. Even if some of the distances may be overestimated, they are not likely to affect the final result after the sampling is repeated M times.

Secondly, the observed average gap $\bar{d}$ in the sampled set is not approximately the same as, but instead proportional to, the desired average gap d in the full data set. The constant of proportionality that connects $\bar{d}$ and d depends on the parameters PT, N, and v, where v is the number of degrees of freedom. In particular, we have shown that the actual relationship between $\bar{d}$ and d can be expressed by the following equation.

$$\bar{d} \approx \left(\frac{PT}{N}\right)^{\frac{1}{v}} d \tag{16}$$

Equation (16) can be explained below: when $N = PT$, $\bar{d} = d$; when the number of samples per degree of freedom is halved but N_j is still so large that the samples are representative, $N = 2^v PT$ and $\bar{d} \approx 2d$. This also implies that under the same distribution with enough points, $(PT)^{1/v}d$ is a constant, which means (3) and (4) are satisfied if we apply (14) to set the smoothing parameter and $v>1$. When (14) is applied and v is 1, (4) is not satisfied, but (4) is not a necessary condition for the estimated PDF to be asymptotically accurate. Our experiments show that our approach works very well on a one-dimensional database.

As we have mentioned above, v means the number of degrees of freedom. For example, when the data points of the dataset are residing on the unit circle in a 2-D space, their attributes x and y can be expressed in terms of the angle of the (x, y) point with respect to the horizontal axis. As a result, in this case, although the data points are two dimensional, they have only one degree of freedom.

Equation (16) is a single equation with two unknowns (that is, d that we want to compute, and v that is unknown). Ideally, two calculations of $\bar{d}$ with different values of sample sizes N would be sufficient to produce the needed value d and the unknown degrees of freedom v. Due to the randomness of the sampling procedure that leads to the computation of d, however, two calculations of $\bar{d}$ are not enough. For the databases we have experimented with, it turned out that five calculations of $\bar{d}$ are sufficient for the calculation of d. To calculate d from the five equations of the form depicted in (16), we utilized a least-square-error procedure.

C. Application to Classification Problems

So far we have discussed the one-class case. For classification problems, there are two ways to set the smoothing parameter:

1) For each class, compute σ_{ij} based on (14), only accessing the patterns in the corresponding class;

2) Apply (14) to all patterns regardless of their classes and use the computed σ_i for all classes.

Although the first method appears more reasonable, we argue that it is less beneficial because each class occupies only a subset of the training points and thus for each class, the estimate of d becomes less accurate after sampling (whose confidence relies on the data size), especially when (15) is violated. Our preliminary experiments verify our argument, although the corresponding results are not listed in this paper.

V. Experiments

A. Experimental Procedures

We compare the following approaches for choosing the smoothing parameters for both PNN and GRNN:

Standard Deviation: set the sigma parameters as the standard deviation for each dimension without discriminating the class labels (which is much better than using the in-class standard deviations in our experiments, although the data are not shown here).

Cross-Validation: the training set is divided equally to a learning set and a validation set. We evaluate $\sigma_i = kSTD(X_i)$ for each attribute X_i, where k is chosen from {1/2, 1/4, 1/8, 1/16}, and $STD(X_i)$ is computed in the previous step. The best k value is selected according to the performance on the validation set. The time for finding the smoothing parameters is defined as the total time for all the runs of PNN/GRNN.

Clustering: run GART to cluster the training set. For PNN, we run GART for each class separately (which is remarkably faster than running GAM on the whole database). The initial standard deviation γ is set to $STD(X_i)$ and the baseline vigilance ρ is set to 0.5 (in fact, we tested higher values of ρ, but they only spent more time without significant improvement in accuracy). It is known that GAM/GART is not sensitive to γ, as long as it is close to the final cluster standard deviation. The parameter ρ controls the size of the clusters: $\rho = 0$ means arbitrarily large clusters are allowed, resulting in only one cluster for GRNN since all regression instances are treated as in the same class; $\rho = 1$ means only zero-sized clusters can be created, reducing to the non-cluster case except when repeated training instances are present. The time for finding the sigma parameters is defined as the time elapsed in GART.

Gap-Based Estimation: apply our approach.

All algorithms, including PNN, GRNN, and GART, are coded efficiently in the same computer language (C/C++) and with the same interface (MATLAB MEX DLL). The recorded time does not include what is spent on file I/O or displaying to the screen. All experiments are carried out in the same and stable software environment.

B. Databases

The databases used in our experiments are listed in Table I. To demonstrate that the standard deviation is not directly related to the optimal sigma value, we created an artificial database *Grass and Trees* that contains only one attribute. The first class is uniformly distributed in [0, 1]. The second class has five clusters, centered at 0, 0.25, 0.5, 0.75, and 1, respectively. Each cluster has also uniform distribution with range 0.05. Both classes occupy 50% of the instances. It can be shown that the Bayes Classifier attains 90% accuracy on this database. Fig. 3 shows that it is difficult to guess the optimal sigma value using the standard deviation, while our

TABLE I
STATISTICS OF DATABASES

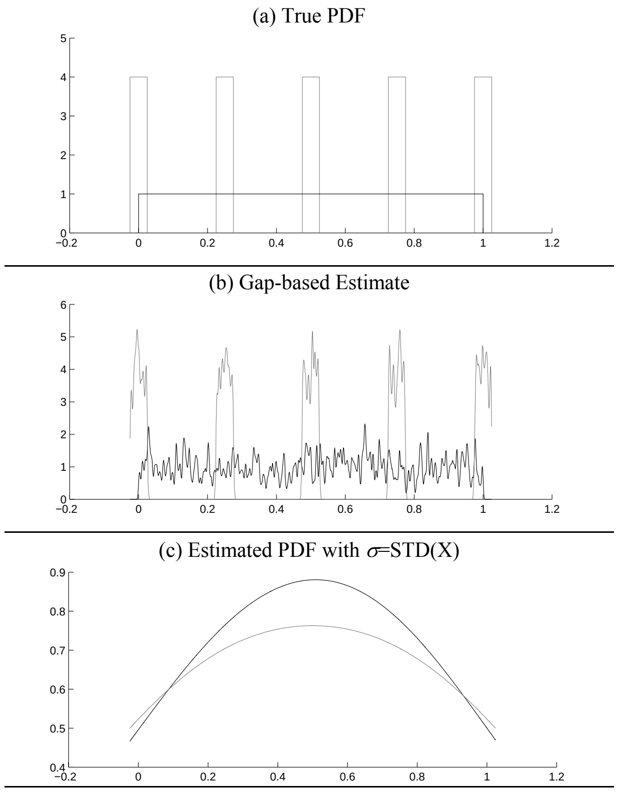
PNN Databases	#Training Points	#Test Points	#Numerical Attributes	#Classes	%Minor Classes [a]
Grass and Trees	2000	4000	1	2	0.5
Modified Iris	500	4800	2	2	0.49812
Segmentation	210	2100	18	7	0.85714
Page Blocks	2000	3473	10	5	0.10193
Abalone	2088	2089	7	3	0.65677
Satellite	4435	2000	36	6	0.7695
Pen Digits	7494	3498	16	10	0.89623
Optical Digits	3823	1797	62	10	0.89872

GRNN Databases	#Training Points	#Test Points	#Numerical Attributes	Output Range [b]	Output Variance [c]
Friedman	400	1000	5	27.116	27.094
Kinematics	4096	4096	8	1.4004	0.06853
Pumadyn	4096	4096	8	24.091	31.41
Bank	4096	4096	8	0.74548	0.023478
Abalone	2088	2089	7	26	10.544
Computer	4096	4096	12	99	348.85

[a]The percentage of the minor classes is 1 minus the percentage of the major training class in the test set. This percentage represents the misclassification rate for the blind classifier, which always predicts the class as the major training class without considering the attributes).

[b]The output range reflects the output in the test set

[c]The output variance is computed as mean($(y-m)^2$), where y is the outputs in the test set and m is the mean value of the outputs in the training set. This variance represents the mean-square-error of the blind predictor (that is, the predictor that always predicts the output as the mean training output without considering the attributes).



Black line: class 1; Gray line: class 2

Figure 3: Estimates of the PDFs for Database "Grass and Trees"

approach produces very accurate estimates. Note that the optimal smoothing parameter can be arbitrarily small when we increase the number of clusters in class 2 while maintaining, at the same time, its overall standard deviation.

The rest of the databases are commonly used benchmark databases. We selected the ones with sufficient size in order to make our results statistically significant. We downloaded all the other classification databases from [12], *Friedman* from [13], and *Kinematics*, *Bank*, and *Computer* from [14]. Following is the brief description of each database:

The *Iris* database represents the classification task of iris plants. We introduced noise to generate enough points, and removed the two attributes with least correlation to the class.

The *Segmentation* database is created from 7 outdoor images of different scenes. 3-by-3 sub-images are manually segmented from the 7 images and are classified with the image features. The third feature in the original database turns out to be constant and thus is removed in our experiments. This database contains only 210 training points, which is a representative test to our sampling methodology.

The *Page Blocks* database consists of the attributes of page layouts in a document with various block types. The major class "text" occupies approximately 90% of the instances.

The *Abalone* database is used for predicting the age of abalones. We removed the categorical attribute in the original database because it is not used on either PNN or GRNN. For PNN, we also grouped the outputs into three classes: 8 and lower, 9-10, 11 and greater, as other researchers have done in the past. The resulting classes, however, are still highly overlapped in the attribute space and current algorithms can attain only approximately 60% accuracy.

The *Satellite* database provides the multi-spectral values extracted from satellite images corresponding to 6 types of soil (previously 7 types, one of which used to be "mixed soil" and was removed due to doubts about its validity).

The *Pen Digits* database stores the information of 250 digits. The attributes are obtained from the coordinates of the points after spatial sampling on the captured trajectories. 30 writers contribute to the training set and 14 to the test set.

The *Optical Digits* database is also concerned with the recognition of handwritten digits, but without temporal information. The images are divided into sub-images and the number of pixels in each sub-image serves as an attribute. The data from 30 writers are used for training and those from the other 13 writers for testing. After two constant attributes are removed from the original database, there are still as many as 62 attributes.

The *Friedman* database is artificial, first used in [15]. The output is defined as $y=10\sin(\pi x_1 x_2)+20(x_3-0.5)^2+10x_4+5x_5+\varepsilon$, where x_1, x_2, x_3, x_4, and x_5 are independent attributes uniformly distributed in [0,1], while ε is Gaussian noise with zero mean and unity variance.

The *Kinematics* database and the *Pumadyn* database are chosen from two families generated from the simulation of two different robot arms. They are concerned with the prediction of the end-effector from a target and the angular

TABLE II
EXPERIMENTAL RESULTS

PNN Databases	Misclassification Rate (%)					Time (seconds)			
	Blind Classifier	Standard Deviation	Cross Validation	Clustering	Gap-based Estimate	Standard Deviation	Cross Validation	Clustering	Gap-based Estimate
Grass and Trees	50.00	38.63	15.60	34.975	10.60	0	0.578	0.094	0.047
Modified Iris	49.81	6.19	5.38	5.1458	5.63	0	0.047	0.063	0.015
Segmentation	85.71	14.38	10.52	20.238	10.81	0	0.031	0.031	0
Page Blocks	10.19	5.79	4.06	36.165	4.78	0	1.156	32.203	0.157
Abalone	65.68	37.10	35.47	40.211	35.38	0	1.015	2.922	0.125
Satellite	76.95	13.00	9.55	13.25	9.70	0.031	16.063	48.328	2.141
Pen Digits	89.62	8.38	2.57	5.8033	3.23	0.032	25.219	22.891	1.938
Optical Digits	89.87	3.28	3.78	78.353	3.62	0.063	16.954	26.5	1.360

(a) PNN Experiments

GRNN Databases	Mean Square Error					Time (seconds)			
	Blind Predicator	Standard Deviation	Cross Validation	Clustering	Gap-based Estimate	Standard Deviation	Cross Validation	Clustering	Gap-based Estimate
Friedman	27.094	9.2244	4.3103	4.9986	4.2921	0	0.047	1.172	0.016
Kinematics	0.06853	0.031563	0.014234	0.019119	0.014304	0.015	5.343	76.187	0.547
Pumadyn	31.41	18.516	14.788	15.068	14.808	0	5.344	83.797	0.531
Bank	0.023478	0.0061562	0.0025063	0.002003	0.0025669	0	5.297	147.33	0.531
Abalone	10.544	6.22	5.1726	5.024	5.5133	0	1.25	18.782	0.156
Computer	348.85	36.455	15.653	333.28	15.764	0.016	7.328	176.67	0.703

(b) GRNN Experiments

acceleration, respectively. For both databases, we selected the non-linear version with 8 attributes and medium noise.

The *Bank* database is generated from a simulator that simulates bank service. The task is to predict the fraction of customers who leave the bank because all queues are full.

The *Computer* database consists of computer activity measures, such as the reading/writing rates. The task is to predict the portion of time that the CPUs run in user mode based on the various data transfer rates.

C. *Experimental Results*

All the results are shown in Table II. For PNN, although the computation of the standard deviation is usually too fast to be timed, it does not appear to be a good value for the smoothing parameters due to its poor accuracy (excluding the blind classifier), especially when the data are distributed in disconnected clusters (see the *Grass and Trees* database).

The cross-validation approach works well in databases where the data corresponding to different classes are small in number and belong to a single connected cluster.

The Clustering approach could work better than cross-validation in defining reasonable smoothing parameters when class data belong to disconnected clusters, but its accuracy is suspect, possibly due to the weak relationship between the optimal smoothing parameter and the cluster standard deviation, as illustrated in Fig. 2. Note that the accuracy is surprisingly low for *Optical Digits*. We examined the results and found that GART output exactly one cluster per training point, due to the high dimensionality, which means the distance among training points tends to be large. The cluster standard deviation is usually too small, because the same point is repeatedly chosen to construct a template.

Our approach always yields an accuracy that is equal or close to the best one. Note that for the *Grass and Trees* database, its accuracy is almost the theoretical optimal one. Moreover, the time spent to produce the smoothing parameters with our approach is only greater than of the standard deviation, and it is scalable to large databases. The experiment also demonstrates that our approach is robust, whether or not the training set is noisy (as *Grass and Trees*, *Modified Iris* and the *Friedman*), small (as *Segmentation*), or high dimensional (which means possible dependency among attributes; see *Optical Digits*).

In the experiments with GRNN, we have exactly the same observations: using the standard deviation is fastest but most inaccurate; cross-validation has a reasonable accuracy while its time is non-scalable; clustering in unstable in accuracy and expensive in time; our approach is the second fastest one with almost best accuracy.

VI. CONCLUSION

In this paper we presented the Gap-based approach of estimating the smoothing parameters for both PNN and GRNN. Our approach was first analyzed for an ideal problem and then applied to more general problems. We utilized sampling techniques to reduce the computational complexity of finding the smoothing parameters to a linear function of the training data size. Our experiments have showed that our proposed approach, the gap-based estimate of the smoothing parameter, is superior to other commonly used approaches, such as cross-validation. It was demonstrated, through these experiments, that the gap-based estimate of the smoothing parameters produced a PNN/GRNN network with good accuracy. The amount of time required to produce the smoothing parameter estimates was of low computationally complexity, and scalable to larger problems.

Appendix – Pseudo Code of Gap-Based Estimation

```
Parameters:
$K_{\max}$: Maximum Repeat Times (natural number). Typical $K_{\max}$=4
$F_M$, $F_N$: Sampling Factor (small positive number). Typical $F_M$=8, $F_N$=4

Main Procedure:
Compute $STD(X_i)$ for $i$=1,2,…,$D$
$M = \min\left(PT, \lfloor F_M\sqrt{PT}\rfloor\right)$
$N = \min\left(PT, \lfloor F_N\sqrt{PT}\rfloor\right)$
$K = \min\left(K_{\max}, \left\lfloor \log_2 \frac{PT}{N}\right\rfloor\right)$
If $K$<1 (which means $PT$ is small) then
        $d$=AverageGap($M$,$PT$)
Else
        $\bar{d}_k$ = AverageGap($M$,$2^kN$)  for $k$=0, 1, 2, … , $K$
        Solve the equations $\frac{1}{v}\log\left(\frac{PT}{2^k N}\right) + \log d = \log \bar{d}_k$
        for $k$=0, 1, 2, … , $K$, treating $\frac{1}{v}$ and $\log d$ as unknowns.
End If
$\sigma_i$=min(4$d$,0.5)$STD(X_i)$ for $i$=1,2,…,$D$

Subroutine AverageGap ($M$, $N$)
For $m = 1,2,...,M$
        Randomly choose a point $\mathbf{X}$.
        For $n = 1,2,...,N$
                Randomly choose a point $\mathbf{X}_n$ different from $\mathbf{X}$.
                $d_{mn}{}^2 = \sum_{i=1}^{D}\left(\frac{X_{in} - X_i}{STD(X_i)}\right)^2$
        End For
        Find the smallest 2$D$ elements from $d_{m1}{}^2, d_{m2}{}^2, ... d_{mN}{}^2$
        Find the largest one (denoted by $d_m{}^2$) in the above elements
End For
Return $\sqrt{\underset{m}{mean}\left[d_m{}^2\right]}$

Note: the smallest 2D elements for each m can be cached so that when
we double N next time, only N more patterns have to be chosen, which
halves the computational complexity.
```

The least-square-error solution to the linear equations can be explicitly given as:

$$\begin{bmatrix} 1/v \\ \log d \end{bmatrix} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{B} \tag{17}$$

$$\mathbf{A} = \begin{bmatrix} \log(PT/N_0) & 1 \\ \vdots & \vdots \\ \log(PT/N_K) & 1 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} \log \bar{d}_0 \\ \vdots \\ \log \bar{d}_K \end{bmatrix} \tag{18}$$

In practice, computing $(A^TA)^{-1}A^TB$ directly is not efficient in both time and space. A recursive algorithm can be applied, which is shown below.

```
$\mathbf{P}_0 = \begin{bmatrix} \log(PT/N_0) & 1 \\ \log(PT/N_1) & 1 \end{bmatrix}$,    $\mathbf{P} = \left(\mathbf{P}_0{}^T\mathbf{P}_0\right)^{-1}$, $\mathbf{Q} = \mathbf{P}_0{}^T\begin{bmatrix} \log \bar{d}_0 \\ \log \bar{d}_1 \end{bmatrix}$
For $k = 2, \ldots, K$
        $\mathbf{a} = [\log(PT/N_k) \quad 1]$
        $\mathbf{P} = \mathbf{P} - \left(\mathbf{Pa}^T\right)\left(\mathbf{aPa}^T + 1\right)^{-1}(\mathbf{aP})$
        $\mathbf{Q} = \mathbf{Q} + \mathbf{a}^T \log \bar{d}_k$
End For
$\begin{bmatrix} 1/v \\ \log d \end{bmatrix} = \mathbf{PQ}$

Note: the update of $\mathbf{P}$ is very simple because $\left(\mathbf{Pa}^T\right)$ is only 2-by-1,
$\left(\mathbf{aPa}^T + 1\right)$ is a scalar, and $(\mathbf{aP})$ is 1-by-2.
```

Acknowledgment

This work was supported in part by a National Science Foundation (NSF) grant CRCD: 0203446. Georgios Anagnostopoulos and Michael Georgiopoulos also acknowledge the partial support from the NSF grant CCLI 0341601.

References

[1] E. Parzen, "On estimation of probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, pp. 1065-1073, 1962.

[2] T. Cacoullos, "Estimation of a multi-variate density," *Annals of the Institute of mathematical Statistics* (Tokyo), Vol. 18, No. 2, pp. 179-189, 1966.

[3] D. F. Specht, "Probabilistic Neural Networks and the Polynomial Adaline as Complementary Techniques for Classification," *IEEE Trans. Neural Networks*, vol.1, no.1, pp.111-121, 1990.

[4] D. F. Specht, "A General Regression Neural Network," *IEEE Trans. Neural Networks*, vol.2, pp.568-576, 1991.

[5] D. F. Specht, "Enhancements to Probabilistic Neural Networks," in *1992 Proc. IJCNN*, vol. 1, pp. 761-768.

[6] D. F. Specht, "Experience with Adaptive Probabilistic Neural Networks and Adaptive General Regression Neural Networks," in *1994 Proc. IEEE World Congress on Computational Intelligence*, vol. 2, pp. 1203-1208.

[7] P. Burrascano, "Learning vector quantization for the Probabilistic Neural Network," *IEEE Tran. Neural Networks*, vol. 2, pp. 458-461, 1991.

[8] H. G. C. Traven, "A neural network approach to statistical pattern classification by 'semi-parametric' estimation of probability density functions," *IEEE Trans. Neural Networks*, vol. 2, pp. 366-377, 1991.

[9] M-L. Tseng, "Integrating Neural Networks with Influence Diagrams for Multiple Sensor Diagnostic Systems," Ph.D. Dissertation, University of California at Berkley, 1991.

[10] J. R. Williamson, "Gaussian ARTMAP: A Neural Network for Fast Incremental Learning of Noisy Multi-Dimensional Maps," *Neural Networks*, vol. 9, no. 5, pp. 881-897, 1996.

[11] J. R. Williamson, "A constructive, incremental-learning network for mixture modeling and classification," *Neural Computation*, vol. 9, pp. 1517-1543, 1997.

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

[13] KEEL (Knowledge Extraction based on Evolutionary Learning) Datasets, Available: http://sci2s.ugr.es/keel-dataset/

[14] Delve (Data for Evaluating Learning in, Valid Experiments), University of Toronto, Toronto, Ontario, Canada. Available: http://www.cs.toronto.edu/~delve/

[15] J. Friedman, "Multivariate adaptive regression splines (with discussion)," *Ann. Stat.*, vol. 19, pp. 1–141, 1991.