LETTER Communicated by Donald Specht

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

Mingyu Zhong
myzhong@ucf.edu
Dave Coggeshall
david.coggeshall@gmail.com
Ehsan Ghaneie
Ehsan.Ghaneie@gmail.com
Thomas Pope
ThomasPope@gmail.com
Mark Rivera
mark.rivera@gmail.com
Michael Georgiopoulos
michaelg@mail.ucf.edu
School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, U.S.A.

Georgios C. Anagnostopoulos
georgio@fit.edu
Department of Electrical and Computer Engineering, Florida Institute of Technology, Melbourne, FL 32901, U.S.A.

Mansooreh Mollaghasemi
mollagha@mail.ucf.edu
Department of Industrial Engineering and Management Systems, University of Central Florida, Orlando, FL 32816, U.S.A.

Samuel Richie
richie@mail.ucf.edu
School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, U.S.A.

Probabilistic neural networks (PNN) and general regression neural networks (GRNN) represent knowledge by simple but interpretable models that approximate the optimal classifier or predictor in the sense of expected value of the accuracy. These models require the specification of an important smoothing parameter, which is usually chosen by cross-validation or clustering. In this article, we demonstrate the problems with the cross-validation and clustering approaches to specify the smoothing parameter, 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. Finally, through experimentation, we show that our approach, referred to as a gap-based estimation approach, is superior in speed to the compared approaches, including support vector machine, and yields good and stable accuracy.

1 Introduction

Classification and regression are two types of common problems in science, technology, and even our daily life. Assuming that the inputs and 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 (Parzen, 1962), extended by Cacoullos (1966), which estimates the class-conditional probabilities as a sum of gaussian functions centered at the training points with appropriately chosen widths (standard deviations), 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 size becomes very large, the approximation becomes equal to the actual class-conditional probabilities. When training data sets are of finite size (as is usually the case in practice), the right choice of smoothing parameters is essential for good performance of the classifier. Probabilistic neural networks (PNN), invented by Specht (1990), approximates a Bayes classifier where the class-conditional probabilities are estimated by using Parzen's approach with gaussian kernels. General regression neural network (GRNNs), also invented by Specht (1991), is the PNN's regression counterpart.

One of the major issues associated with the PNN and GRNN is how to choose the smoothing parameter(s) involved in the gaussian functions utilized to estimate the conditional probabilities. Unlike linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) or expectation-maximization algorithms where the number of gaussian functions is preset to a small number, PNN or GRNN applies a variable number of kernels (one per each training example), and consequently the maximum likelihood estimate of the smoothing parameter is zero, resulting in an estimate of the class-conditional probabilities as a series of impulses, thus affecting negatively the generalization. Specht (1992, 1994) suggested using cross-validation to estimate the smoothing parameters. This approach, although simple, has two major drawbacks. The first is that 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 (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. The second drawback is that cross-validation is very time-consuming. The time complexity of cross-validation in PNN or 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 $O(N{\cdot}D{\cdot}PT{\cdot}PV)$.

To overcome the first difficulty, Galleske and Castellanos (2002) proposed the ellipsoidal model to estimate the smoothing parameters. However, their approach is based on the most complicated PNN structure: each instance corresponds to an individual covariance matrix as the smoothing parameter. To optimize the matrix for each instance, one has to find out the "closest" instance belonging to another class. Therefore, the time complexity issue is not solved but rather worsened.

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 for the PNN neural network are the learning vector quantization (LVQ) approach (Burrascano, 1991), K-means clustering (Traven, 1991), mixture of gaussians (Tseng, 1991), and the dynamic decay adjustment (DDA) approach (Berthold, 1998). These approaches modify PNN by reducing the number of kernels. For example, the DDA approach actually optimizes the smoothing parameter for radial basis function (RBF) networks. In this article, we stick with the PNN model (one kernel per instance) and attempt to find out whether these clustering approaches can simplify the optimization of the smoothing parameters of PNN significantly. We apply an unsupervised variant of gaussian ARTMAP (GAM) (Williamson, 1996, 1997), which is similar to RBF, 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 article.

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 is also an advantage in practical situations, where the data set given is

small, and we do not have the luxury of defining a sizable cross-validation set.

The organization of the letter is as follows. Section 2 introduces PNN and GRNN, including their clustered versions. Section 3 discusses the effect of the smoothing parameter on PNN and GRNN. Section 4 describes our approach of choosing the smoothing parameter. Section 5 compares our approach of choosing the smoothing parameter to other commonly used approaches. Section 6 provides a summary of our work and concluding remarks. For the rest of the article, we assume that the reader is familiar with GAM.

2 Preliminaries

2.1 Probabilistic Neural Network. The Bayes classifier is based on the following formula for finding the class to which a datum $\mathbf{x}$ belongs:

$$P(c_j|\mathbf{x}) = \frac{f(\mathbf{x}|c_j)P(c_j)}{f(\mathbf{x})}. \tag{2.1}$$

The above probabilities for every class j in our pattern classification task are 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 by Parzen (1962). In particular, the following approximation is utilized to estimate these class-conditional probabilities (see Specht, 1990):

$$f(\mathbf{x}|c_j) = \frac{1}{(2\pi)^{D/2}\sigma^D PT_j} \sum_{r=1}^{PT_j} \exp\left[-\frac{(\mathbf{x}-\mathbf{X}_r^j)(\mathbf{x}-\mathbf{X}_r^j)}{2\sigma^2}\right], \tag{2.2}$$

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 rth such training pattern, $\mathbf{x}$ is the input pattern to be classified, and σ is the smoothing parameter. Cacoullos (1966) points out that this estimation will approach asymptotically the real probability density function (PDF) if both of the following equations are true:

$$\lim_{PT_j \to \infty} \sigma = 0 \tag{2.3}$$

$$\lim_{PT_j \to \infty} PT_j \sigma = \infty. \tag{2.4}$$

Equation 2.2 is the basis of the PNN classifier. The smoothing parameter σ should be selected properly, as further discussed in section 3. In general, the smoothing parameters may depend on the dimension and the class of the data. For simplicity, we assume attribute independence, 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

$$f(\mathbf{x}|c_j) = \frac{1}{(2\pi)^{D/2}\prod_{i=1}^{D}\sigma_{ij}PT_j}\sum_{r=1}^{PT_j}\exp\left[-\sum_{i=1}^{D}\frac{\left(x_i - X_{ir}^{j}\right)^2}{2\sigma_{ij}^2}\right], \tag{2.5}$$

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

2.2 General Regression Neural Network. 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:

$$E(y|\mathbf{x}) = \frac{\int_{-\infty}^{\infty} yf(\mathbf{x}, y)dy}{\int_{-\infty}^{\infty} f(\mathbf{x}, y)dy}. \tag{2.6}$$

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

$$f(\mathbf{x}, y) = \frac{\sum_{r=1}^{PT}\exp\left[-\frac{(y-y_r)^2}{2\sigma_0^2} - \sum_{i=1}^{D}\frac{(x_i-X_{ir})^2}{2\sigma_i^2}\right]}{PT(2\pi)^{(D+1)/2}\prod_{i=0}^{D}\sigma_i}. \tag{2.7}$$

Equations 2.6 and 2.7 can be used to show that $E(y|\mathbf{x})$ can be estimated by

$$E(y|\mathbf{x}) = \frac{\sum_{r=1}^{PT} y_r \exp\left[-\sum_{i=1}^{D}\frac{(x_i-X_{ir})^2}{2\sigma_i^2}\right]}{\sum_{r=1}^{PT}\exp\left[-\sum_{i=1}^{D}\frac{(x_i-X_{ir})^2}{2\sigma_i^2}\right]}, \tag{2.8}$$

where σ_i is the smoothing parameter across dimension i.

2.3 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 or GRNN, we should expect the smoothing parameter to depend on the clusters. When clustering is applied to the PNN training data, it is not

difficult to see that the modified version of equation 2.5 is

$$F(\mathbf{x}|c_j) = \frac{\sum_{r=1}^{N^j} \frac{N_r^j}{\prod_{i=1}^{D} \sigma_{ir}^j} \exp\left[-\sum_{i=1}^{D} \frac{\left(x_i - \bar{X}_{ir}^j\right)^2}{2\sigma_{ir}^2}\right]}{(2\sigma)^{D/2} \sum_{r=1}^{N_j} N_{jr}}, \tag{2.9}$$

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 $\bar{X}_{ir}^j$ are the smoothing parameter and the mean value of the ith dimension for the rth cluster in class j, respectively.

Similarly, the GRNN equation, when clustering is applied to the GRNN training data, becomes

$$E(y|\mathbf{x}) = \frac{\sum_{r=1}^{PT} \frac{y_r N_r}{\prod_{i=1}^{D} \sigma_{ir}} \exp\left[-\sum_{i=1}^{D} \frac{(x_i - \bar{X}_{ir})^2}{2\sigma_{ir}^2}\right]}{\sum_{r=1}^{PT} \frac{N_r}{\prod_{i=1}^{D} \sigma_{ir}} \exp\left[-\sum_{i=1}^{D} \frac{(x_i - \bar{X}_{ir})^2}{2\sigma_{ir}^2}\right]}. \tag{2.10}$$

3 Analysis of the Smoothing Parameter in the One-Dimensional Cases

3.1 Nonclustered Cases. For simplicity, let us first consider only one dimension and one class without clustering. In this case, equation 2.5 reduces to

$$f(x) = \frac{1}{(2\pi)^{1/2}\sigma\, PT} \sum_{r=1}^{PT} \exp\left[-\frac{(x - X_r)^2}{2\sigma^2}\right]. \tag{3.1}$$

It is not difficult to prove that

$$\mathrm{VAR}(\hat{X}) = \mathrm{VAR}(X) + \sigma^2, \tag{3.2}$$

where the left-hand side represents the expected value of the variance of the point x using the estimated PDF given in equation 3.1, 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

$$\sigma \ll STD(X), \tag{3.3}$$

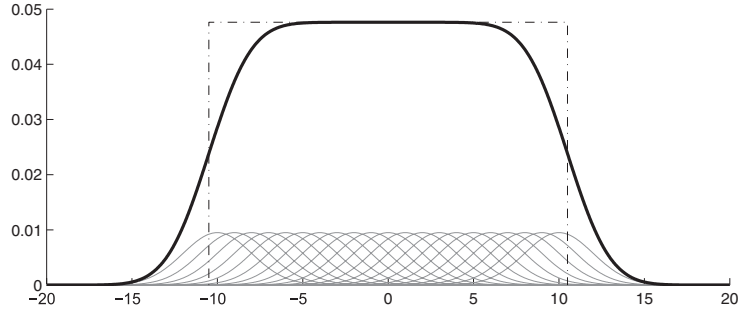
where $STD(X)$ is the unbiased standard deviation of X. This conclusion agrees with equation 2.3, since the standard deviation usually approaches a positive constant number when the number of instances goes to infinity.

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

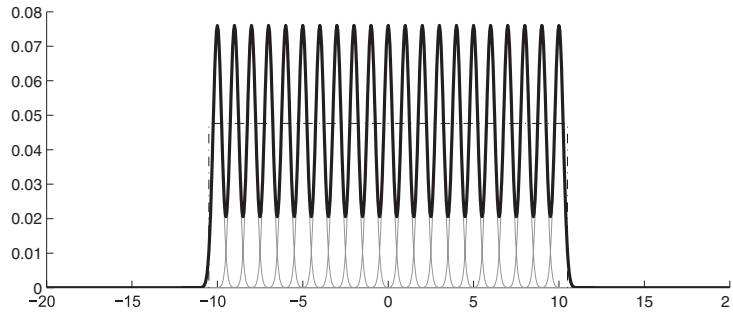
space. This could be beneficial for removing irrelevant attributes for classification, but it alters the PDFs, while our goal in this article is to accurately estimate the PDFs, which can be used by algorithms other than PNN or GRNN. On the other hand, σ_i cannot be too small; otherwise, the PDF becomes spiky (consisting of a number of impulse functions) at the training points and is almost zero at other places, which causes the overtraining problem.

To understand the effect of the smoothing parameter better, consider an example where the training set is $\{-10, -9, \ldots, 10\}$ 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, which will be further discussed in section 4. Figure 1 shows the resulting estimated PDF in three typical cases. In Figure 1a, the smoothing parameter is properly set, and the estimated PDF is almost the same as the true one except near the two bounds; for better generalization, we do not require the PDF to drop too quickly to 0 outside the range $[-10.5, 10.5]$ (see Figure 4b). In Figure 4b, the smoothing parameter is too small, and the estimated PDF consists of spikes. Although in this case the PDF outside the range $[-10.5, 10.5]$ is accurate and we will have few false positives, the PDF is too small inside the range but off any training point. In a multidimensional space, once a test point is not close enough to any training point, the estimated PDF is close to zero for all classes, and the decision of the PNN is not reliable. In Figure 4c, the smoothing parameter is too large, which means all the training points have significant influence on the PDF at the test point even if they are far away from the test point. When the true PDFs of two classes are overlapped, the crossing points of the discriminant functions tend to be shifted from the true ones (consider the extreme case $\sigma = \infty$, which shifts the crossing points to ∞, and the decision will be based on prior probabilities only). This figure also verifies our previous statements. It shows that the standard deviation of the observed attribute values is usually too large to be used as the smoothing parameter. In appendix B, we prove that to estimate the norms accurately, we should set the smoothing parameter much smaller than the standard deviation.

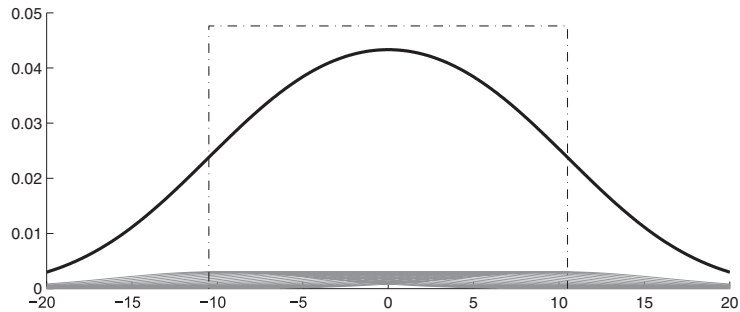
3.2 Clustered Case. Suppose the previous training set with 21 instances is clustered, with each cluster containing the same number of instances. Figure 2 shows the effect of the cluster size and the smoothing parameter on the PDF estimation. In this example, neither the cluster standard deviation nor a reasonable value for the smoothing parameter yields an accurate PDF for both the three-clustered and the seven-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.



(a) Reasonable Smoothing Parameter (σ=2)

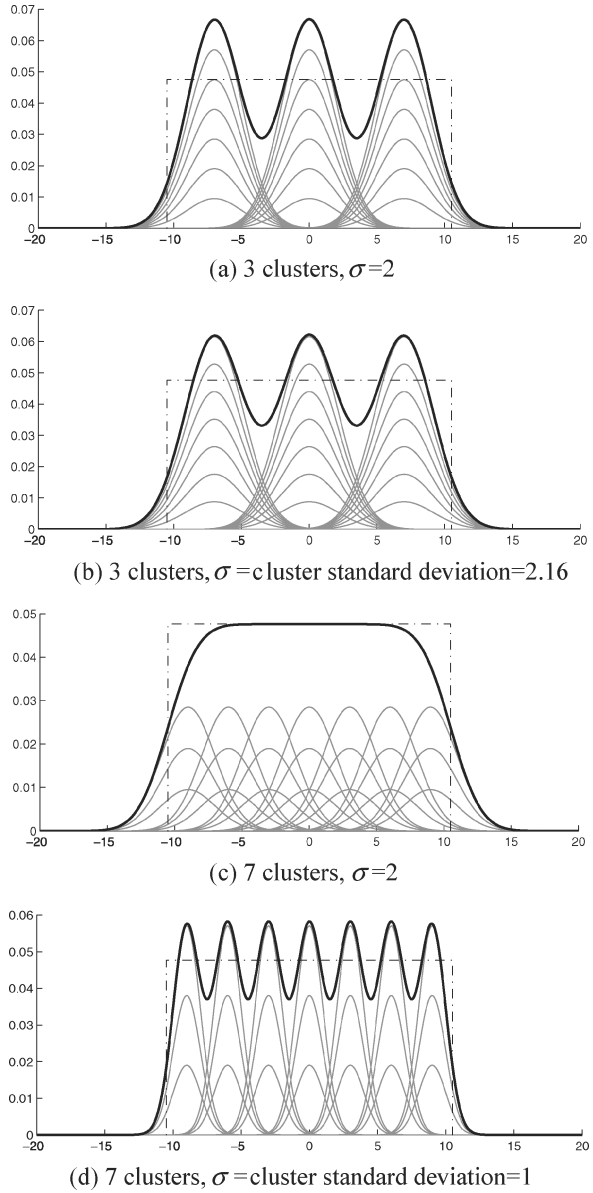


(b) Small Smoothing Parameter (σ=0.25)



(c) Large Smoothing Parameter (σ=Standard Deviation=6.2)

Figure 1: Estimates of PDF with various σ values (no clustering). Dash-dot line: desired (true) PDF. Gray solid line: gaussian function centered at a training point. Black solid line: estimated PDF.



(a) 3 clusters, σ=2

(b) 3 clusters, σ =cluster standard deviation=2.16

(c) 7 clusters, σ=2

(d) 7 clusters, σ =cluster standard deviation=1

Figure 2: Estimates of PDF with various σ values (with clustering). 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.

4 Our Approach: Gap-Based Estimation

4.1 Result from a Simple Model. Figure 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, originates from the following simple idea: when the training points are equally spaced in $(-\mathrm{L}, 0]$ where $\mathrm{L} \gg 1$, the estimated PDF should (1) be almost constant at points located at the left of the origin and (2) drop to almost zero at the points located at the right of the origin (we allow, though, 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 (see appendix C). In practice, the training points are seldom equally spaced even if uniformly distributed. In this case, we have to generalize the definition of d. If we define d as the average distance between a point and its nearest neighbor (i.e., $\sum_i \min_{j \neq i} |x_i - x_j|/N$), it is easy to prove that $d \leq d_0 = (\max_i x_i - \min_i x_i)/(N-1)$, where d_0 is our desired value if the points are uniformly distributed in 1D space (we cannot apply d_0 directly to multidimensional spaces). This indicates that the observed d is usually smaller than our desired value. It is also difficult to provide an analytical expression to adjust d in a general case. To address this problem, first we double the bounds as $1.38d < \sigma < 4.36d$ and choose $\sigma = 4d$ for better generalization. Second, we define d as the average value of the largest distances between a point and its 2D nearest neighbors. If this method is applied to our starting ideal model, the computed d is exactly the same.

4.2 Sampling in Multidimensional Space. For multidimensional cases, we should standardize each dimension first so that each dimension has unity standard deviation before computing the gap, which prevents bias toward dimensions with large data. In these cases,

$$\sigma_i = \min(4d, 0.5) STD(X_i), \tag{4.1}$$

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 equation 2.5 is violated. In this case, we replace $4d$ to 0.5 to retain the validity of inequality 3.3.

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 $M \ll PT$ when PT is large, and for each one ($\mathbf{X}$) of these points, we calculate the largest value of the minimum 2D distances to another small

sample of N points, where $N \ll PT$ 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 from the $\mathbf{X}$, 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,

$$PT \geq 4C^2 \tag{4.2}$$

In this case, if we set N to be equal to $4\sqrt{PT}$, we can prove that the probability that none of the N sampled points are in the same cluster as $\mathbf{X}$ does not exceed $e^{-8} = 3.35 \times 10^{-4}$. 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.

Second, the observed average gap δ 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 δ and d depends on the parameters PT, N, and v, where v is the number of intrinsic dimensionality. In particular, we have shown that the actual relationship between δ and d can be expressed by the following equation:

$$\delta \approx \left(\frac{PT}{N}\right)^{1/v} d. \tag{4.3}$$

Equation 4.3 can be explained below: when $N = PT$, $\delta = d$; when the number of samples per intrinsic dimension is halved but N_j is still so large that the samples are representative, $N = 2^{-v}PT$ and $\delta \approx 2d$. This also implies that under the same distribution with enough points, $(PT)^{1/v}d$ is a constant, which means equations 2.3 and 2.4 are satisfied if we apply equation 4.1 to set the smoothing parameter and $v > 1$. When equation 4.1 is applied and v is 1, equation 2.4 is not satisfied, but it is not a necessary condition for the estimated PDF to be asymptotically accurate. Our experiments show that our approach works very well for one-dimensional databases.

As we mentioned above, v means the intrinsic dimensionality. For example, when the data points of the data set are residing on the unit circle in a 2D 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 intrinsic dimension.

Equation 4.3 is a single equation with two unknowns (that is, d, which we want to compute, and v, which is unknown). Ideally, two calculations of δ with different values of sample sizes N would be sufficient to produce

the needed value d and the unknown intrinsic dimensionality v. Due to the randomness of the sampling procedure that leads to the computation of d, however, two calculations of δ are not enough. For the databases we have experimented with, it turned out that five calculations of δ are sufficient for the calculation of d. To calculate d from the five equations of the form depicted in equation 4.3, we utilized a least-square-error procedure.

4.3 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 equation 4.1, by using only the patterns in the corresponding class.
2. Apply equation 4.1 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 equation 4.2 is violated. Our preliminary experiments verify our argument, although the corresponding results are not set out in this article.

5 Experiments

5.1 Experimental Procedures. We compare the following approaches for choosing the smoothing parameters for both PNN and GRNN:

- **Standard deviation**: Set the smoothing 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 justification for this statement is not included in this article).
- **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 as explained in the previous approach. 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 (see section 1) 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 required more time to

converge, without significantly improving the attained 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 noncluster case except when repeated training instances are present. The time for finding the smoothing parameters is defined as the time required for GART to converge.

- **Gap-based estimation**: Our approach is described in section 4.

In addition, we compared PNN with C-SVC (support vector classifier) and GRNN with epsilon-SVR (support vector regressor). We applied the radial basis function (RBF) kernel for both SVC and SVR as most other researchers have done in earlier support vector machine (SVM) publications. The implementations of SVC and SVR were downloaded from Chang and Lin (2001), where the authors suggest grid search to optimize the parameters (for loose grid search, $C = \{2^{-5}, 2^{-4}, \ldots, 2^{15}\}$ and gamma $= \{2^{-15}, 2^{-14}, \ldots, 2^{3}\}$). We followed these suggestions and found that using such a wide range of each parameter is reasonable, because among the optimal parameters in our experiments (one per database), min $C = 2^{-5}$, max $C = 2^{15}$, min gamma $= 2^{-15}$, and max gamma $= 2$.

All algorithms, including PNN, GRNN, GART, SVC, and SVR, 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 input and output or displaying to the screen. All experiments are carried out in the same and stable software environment.

5.2 Databases. The databases used in our experiments are listed in Table 1. We used only large databases because they guarantee statistical significance of the results presented here. Therefore, we did not use the small databases as in Galleske and Castellanos (2002) and Berthold (1998).

To demonstrate that the standard deviation is not directly related to the optimal smoothing 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. Figure 3 shows that it is difficult to guess the optimal smoothing value using the standard deviation, while our 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 its overall standard deviation.

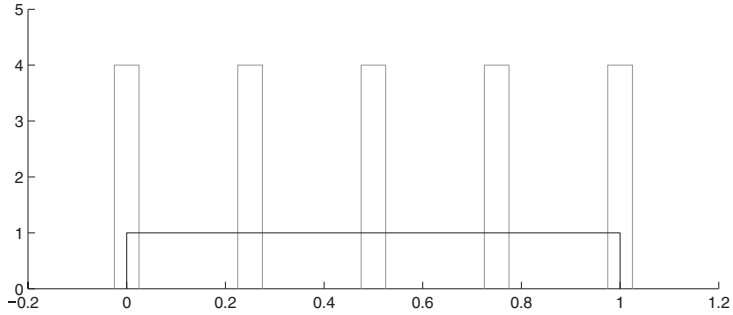
Table 1: Statistics of Databases.

PNN Databases	Number of Training Points	Number of Test Points	Number of Numerical Attributes	Number of Classes	% Minor Classes[a]
Grass and Trees	2000	4000	1	2	50.000
Modified Iris	500	4800	2	2	49.812
Segmentation	210	2100	18	7	85.714
Page Blocks	2000	3473	10	5	10.193
Abalone	2088	2089	7	3	65.677
Satellite	4435	2000	36	6	76.950
Pen Digits	7494	3498	16	10	89.623
Optical Digits	3823	1797	62	10	89.872
GRNN Databases	**Number of Training Points**	**Number of Test Points**	**Number of 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

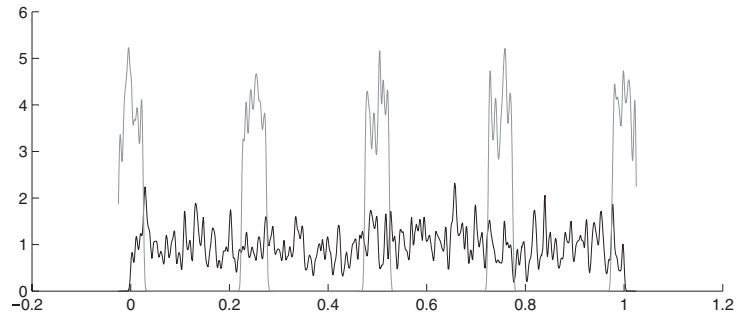
[a] The percentage of the minor classes is 1 minus the percentage of the major training class in the test set. This percentage represents the misclassification rate for the blind classifier, which always predicts the class as the major training class without considering the attributes).
[b] The output range reflects the output in the test set.
[c] The output variance is computed as mean$((y - m)^2)$, where y is the output 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).

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 Newman, Hettich, Blake, and Merz (1998), Friedman from KEEL (2002), and Kinematics, Bank, and Computer from the Delve Repository (Rasmussen et al., 1996). Following is a brief description of each database:

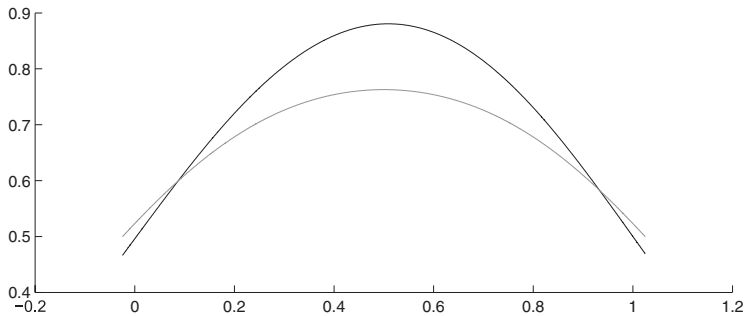
- The Iris database represents the classification task of iris plants. We introduced noise to generate enough points and removed the two attributes with the least correlation to the class label.
- The Segmentation database is created from 7 outdoor images of different scenes. Three-by-three subimages are manually segmented from the 7 images and are classified with the image features. The original database contains 19 attributes, but the third attribute turns out to be constant and thus is removed in our experiments. This database



(a) True PDF



(b) Gap-based Estimate



(c) Estimated PDF with $\sigma = \mathrm{STD(X)}$

Figure 3: Estimates of the PDFs for Grass and Trees database. Black line: class 1; gray line: class 2.

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, and 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 multispectral values extracted from satellite images corresponding to six types of soil (previously seven 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. Thirty 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 subimages, and the number of pixels in each subimage 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, and was first used by Friedman (1991). 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 acceleration, respectively. For both databases, we selected the nonlinear version with eight 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 and 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.

5.3 Experimental Results. All the results are shown in Table 2. 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. However, since the number of candidate values is very limited, cross-validation is sometimes risky, as shown in the experiments with the Grass and Trees database.

The clustering approach tested here 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 Figure 2. Note that the accuracy is surprisingly low for Optical Digits. We examined the results and found that GART outputs 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. Of course, these results reflect only the GART approach; one could improve the clustering approach by using other clustering algorithms or more complicated estimation of the smoothing parameters for each cluster.

Our approach always yields an accuracy that is equal or close to the best one. Note that the accuracy of the Grass and Trees database is almost the theoretical optimal one. Moreover, the time spent to produce the smoothing parameters with our approach is an order of magnitude faster than the cross-validation approach, and it is scalable to large databases (see Figure 4). The experiments also demonstrate 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 our experiments, the SVM accuracy is always better than the gap-based estimate accuracy (except for the Grass and Trees problem), but not significantly better in most instances. The time that it takes to choose the correct SVM parameters is in most cases three to four orders of magnitude larger than the time required by the gap-based estimate approach for choosing PNN parameters, which leads us to the obvious conclusion that there is merit in using the gap-based estimate PNN classifier or GRNN regressor.

Table 2: Experimental Results.

	Misclassification Rate (%)						Time (seconds)			
PNN Databases	Blind Classifier	SVC	Standard Deviation	Cross Validation	Clustering	Gap	SVC	Cross-Validation	Clustering	Gap
Grass and Trees	50.00	11.53	38.63	15.60	34.98	10.60	2.34E3	0.578	0.094	0.047
Modified Iris	49.81	5.50	6.19	5.38	5.15	5.63	7.52E1	0.047	0.063	0.015
Segmentation	85.71	9.52	14.38	10.52	20.24	10.81	1.66E1	0.031	0.031	0
Page Blocks	10.19	3.54	5.79	4.06	36.17	4.78	5.24E2	1.156	32.203	0.157
Abalone	65.68	33.32	37.10	35.47	40.21	35.38	6.69E3	1.015	2.922	0.125
Satellite	76.95	8.40	13.00	9.55	13.25	9.70	1.02E4	16.063	48.328	2.141
Pen Digits	89.62	1.94	8.38	2.57	5.80	3.23	1.95E4	25.219	22.891	1.938
Optical Digits	89.87	2.78	3.28	3.78	78.35	3.62	8.86E3	16.954	26.5	1.360
	Mean Square Error						Time (seconds)			
GRNN Databases	Blind Regressor	SVR	Standard Deviation	Cross Validation	Clustering	Gap	SVR	Cross Validation	Clustering	Gap
Friedman	27.094	1.551	9.224	4.310	4.999	4.292	2.00E3	0.047	1.172	0.016
Kinematics	0.069	0.0069	0.032	0.014	0.019	0.014	1.21E5	5.343	76.187	0.547
Pumadyn	31.410	11.10	18.516	14.788	15.068	14.808	1.24E5	5.344	83.797	0.531
Bank	0.023	0.00135	0.006	0.003	0.002	0.003	2.11E3	5.297	147.330	0.531
Abalone	10.544	4.59	6.220	5.173	5.024	5.513	5.49E4	1.250	18.782	0.156
Computer	348.85	10.88	36.455	15.653	333.28	15.764	1.23E5	7.328	176.670	0.703

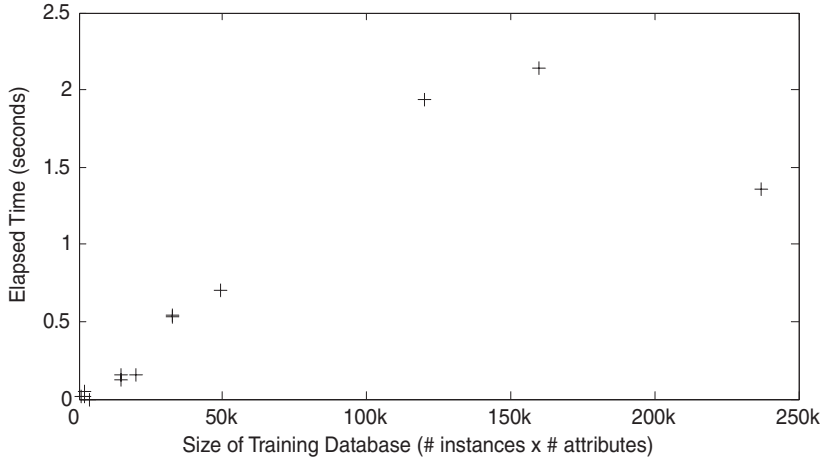


Figure 4: Elapsed time of gap-based estimation versus database size.

In the experiments with GRNN, we have exactly the same observations: using the standard deviation is fastest but has the worst accuracy; cross-validation has reasonable accuracy while its time is nonscalable; the tested clustering approach using cluster standard deviations as smoothing parameters is unstable in accuracy and expensive in time; our approach is the fastest one (excluding the unreliable STD approach) with good accuracy; SVM has the best accuracy, but it requires the longest time to optimize its parameters.

6 Conclusion

In this letter, 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 showed that our proposed approach, the gap-based estimate of the smoothing parameter, is faster than other commonly used approaches for finding the smoothing parameters, such as cross-validation. It was also demonstrated, through experimentation, that the gap-based estimate of the smoothing parameters produced a PNN/GRNN network with good and stable accuracy, comparable (albeit inferior) to the accuracy achieved by a SVM approach, but achieving this accuracy in orders of magnitude faster than the SVM approach. The computational complexity required by the gap-based estimate approach to produce the smoothing

parameter estimates in PNN/GRNN was low and scalable to larger problems.

Appendix A: Pseudo-Code of Gap-Based Estimation

The parameters are

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

Subroutine AverageGap (*M, N*)

For $m = 1, 2, \ldots, M$
  Randomly choose a point $\mathbf{X}$.
For $n = 1, 2, \ldots, N$
  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 $2D$ elements from $d_{m1}^2, d_{m2}^2, \ldots, d_{mN}^2$
  Find the largest one (denoted by d_m^2) in the above elements
End For
Return $\sqrt{\underset{m}{mean}[d_m^2]}$

Note that 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:

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

$$\mathbf{A} = \begin{bmatrix} \log(PT/N_0) & 1 \\ \vdots & \vdots \\ \log(PT/N_K) & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \log \bar{d}_0 \\ \vdots \\ \log \bar{d}_k \end{bmatrix}.$$

In practice, computing $(\mathrm{A}^T\mathrm{A})^{-1}\mathrm{A}^T\mathrm{B}$ directly is not efficient in either time or space. A recursive algorithm can be applied, which is shown below:

$$\mathbf{P}_0 = \begin{bmatrix} \log(PT/N_0) & 1 \\ \log(PT/N_1) & 1 \end{bmatrix}, \quad \mathbf{P} = \left(\mathbf{P}_0^T\mathbf{P}_0\right)^{-1}, \quad \mathbf{Q} = \mathbf{P}_0^T \begin{bmatrix} \log \bar{d}_0 \\ \log \bar{d}_1 \end{bmatrix}$$

For $k = 2, \ldots, K$
  $\mathbf{a} = [\log(PT/N_k)\ 1]$
  $\mathbf{P} = \mathbf{P} - (\mathbf{P}\mathbf{a}^T)(\mathbf{a}\mathbf{P}\mathbf{a}^T + 1)^{-1}(\mathbf{a}\mathbf{P})$
  $\mathbf{Q} = \mathbf{Q} + \mathbf{a}^T \log \bar{d}_k$
End For

$$\begin{bmatrix} 1/v \\ \log d \end{bmatrix} = \mathbf{P}\mathbf{Q}$$

Note that the update of $\mathbf{P}$ is very simple because $(\mathbf{P}\mathbf{a}^T)$ is only 2-by-1, $(\mathbf{a}\mathbf{P}\mathbf{a}^T + 1)$ is a scalar, and $(\mathbf{a}\mathbf{P})$ is 1-by-2.

Appendix B: Relationship Between σ and STD

Consider the one-dimensional case. The PDF is estimated as

$$p(x|c_j) = \frac{1}{\sqrt{2\pi}\sigma_j PT_j} \sum_{r=1}^{PT_j} \exp\left[-\frac{\left(x - X_r^j\right)^2}{2\sigma_j^2}\right].$$

Let $f_{\hat{X}}(x) = p(x|c_j)$ stand for the estimated PDF and $f_X(x)$ represent the real unknown PDF of the attribute X^j of class j. When PT_i is sufficiently

large,

$$f_{\hat{X}}(x) = \frac{1}{\sqrt{2\pi}\sigma_j} E\left\{\exp\left[-\frac{(x - X^j)^2}{2\sigma_j^2}\right]\right\}$$
$$= \frac{1}{\sqrt{2\pi}\sigma_j}\int_{-\infty}^{\infty}\exp\left[-\frac{(x-y)^2}{2\sigma_j^2}\right] f_X(y)dy,$$

where $E[X]$ denotes the expected value of the random variable X. It can be easily proven that

$$E[\hat{X}] = E[X^j], \quad VAR[\hat{X}] = VAR[X^j] + \sigma_j^2.$$

Therefore, the estimated PDF is not the real PDF since the variance is different, unless σ_j is zero, which always overfits the network to the training set. In fact, if we seek another kernel function $F(x, X)$ such that $f_X(x) = E[F(x, X)]$ for any x, the only solution is

$$F(x, X) = \delta(x - X) = \lim_{\sigma\to 0}\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-X)^2}{2\sigma^2}\right].$$

This observation leads to constraint 2.3. It also suggests that $\sigma_j^2 \ll VAR[X^j]$, or $\sigma_j \ll STD[X^j]$. The above result is also applicable to the multidimensional case.

Appendix C: Relationship Between σ and d

Now we assume that the attributes have been standardized within each class, that is, $STD[X^j]$ is always one. An important requirement for the σ value of a single class can be derived from a simple model. Assume that the patterns in this class are uniformly distributed within a D-dimensional hypercube whose range in every dimension is $(-L, L)$, and the training data set contains $PT = (2m+1)^D$ (the subscript j is omitted since only this class is discussed) exemplar points regularly located as an array in the hyperbox, as shown in Figure 5.

The figure reflects the case where $m = 4$, $D = 2$ for simplicity. Actually m and n can be much larger. One of the points is $(0, 0, 0, \ldots, 0)$ and one of its neighbors is $(d, 0, 0, \ldots, 0)$, where $d = 2L/(2m+1)$. This model may seem too ideal, but it will be shown that the results can be applied to more general cases.

Consider the center P in a "cell" box that has the following 2^D corners $(0, 0, 0, \ldots, 0), (d, 0, 0, \ldots, 0), (0, d, 0, \ldots, 0), \ldots (d, d, \ldots, d)$. Obviously $P = (d/2, d/2, \ldots, d/2)$. Since the points are uniformly distributed, it is expected

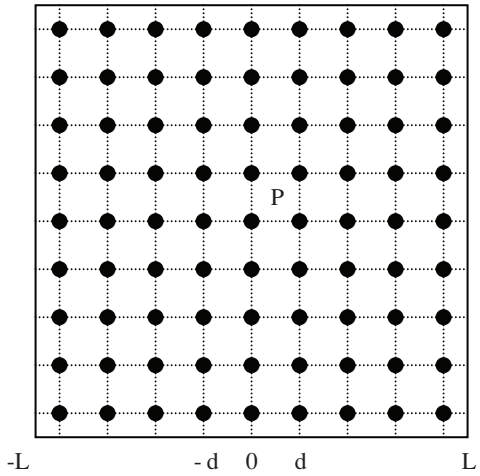


Figure 5: A simple model.

that the estimated PDF at P is the same as that at the origin:

$$\frac{1}{\sigma^D}\sum_{i_1=-m}^{m}\sum_{i_2=-m}^{m}\cdots\sum_{i_n=-m}^{m}\exp\left[-\frac{1}{2\sigma^2}\sum_{l=1}^{D}\left(i_l d-\frac{d}{2}\right)^2\right]$$
$$=\frac{1}{\sigma^D}\sum_{i_1=-m}^{m}\sum_{i_2=-m}^{m}\cdots\sum_{i_n=-m}^{m}\exp\left[-\frac{1}{2\sigma^2}\sum_{l=1}^{D}(i_l d)^2\right].$$

After some calculations, the above equality leads to the following equation:

$$\sum_{i=-\infty}^{\infty}k^{(i-0.5)^2}=\sum_{i=-\infty}^{\infty}k^{i^2}\ \text{where}\ k=\exp\left(-\frac{d^2}{2\sigma^2}\right).$$

Numerical solution of this equation yields $\sigma^2 \geq 0.476d^2$.

Now consider a similar model in which the training patterns are distributed in a hypercube with half volume as the previous one, but with the same density. That is, all the patterns with positive x_1 (coordinate in the first dimension) are removed. Then consider the point $Q = (q \times d, 0, 0, \ldots, 0)$ where q is positive. The estimated PDF at Q should be small enough because Q is outside the region where the training patterns are distributed. With similar derivations, the following condition should be met:

$$0.476d^2 \leq \sigma^2 \leq 4.58d^2.$$

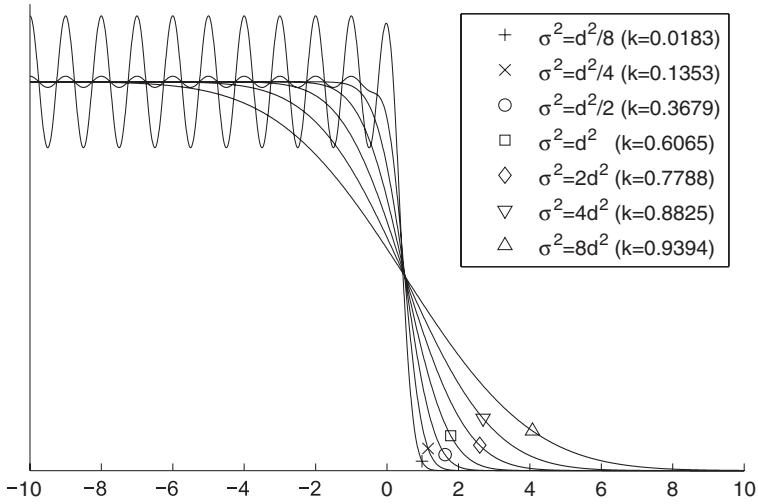


Figure 6: Estimated PDF with various σ^2.

Thus, the smoothing parameter is allowed to vary in a certain way. To show this more clearly, we plot the PDF with various σ^2 in Figure 6, assuming the training instances are $\{-50, -49, \ldots, 0\}$ ($d^2 = 1$).

Acknowledgments

This work was supported in part by the NSF grants: CRCD: 0203446, CCLI: 0341601, DUE: 05254209, IIS-REU: 0647120, and IIS-REU: 0647018.

References

Berthold, M. R. (1998). Constructive training of probabilistic neural networks. *Neurocomputing, 19*, 167–183.

Burrascano P. (1991). Learning vector quantization for the probabilistic neural network. *IEEE Transactions on Neural Networks, 2*, 458–461.

Cacoullos, T. (1966). Estimation of a multi-variate density. *Annals of the Institute of Mathematical Statistics, 18*(2), 179–189.

Chang, T., & Lin, T. (2001). *LIBSVM: A library for support vector machines*. Available online at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Friedman, J. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics, 19*, 1–141.

Galleske I., & Castellanos, J. (2002). Optimization of the kernel functions in a probabilistic neural network analyzing the local pattern distribution. *Neural Computation, 14*, 1183–1194.

KEEL (Knowledge Extraction Based on Evolutionary Learning) data sets. (2002). Available online at http://www.keel.es/.

Newman, D. J., Hettich, S., Blake, C. L., & Merz, C. J. (1998). *UCI Repository of machine learning databases*. Irvine: Department of Information and Computer Science, University of California, Irvine. Available online at http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Parzen, E. (1962). On estimation of probability density function and mode. *Annals of Mathematical Statistics, 33*, 1065–1073.

Rasmussen, C. E., Neal, R. M., Hinton, G. E., Camp, D. van, Revow, M., Ghahramani, Z., Kustra, R., & Tibshirani, R. (1996). *The DELVE manual*. Available online at http://www.cs.toronto.edu/∼delve/.

Specht, D. F. (1990). Probabilistic neural networks and the polynomial Adaline as complementary techniques for classification. *IEEE Transactions on Neural Networks, 1*(1), 111–121.

Specht, D. F. (1991). A general regression neural network. *IEEE Transactions on Neural Networks, 2*, 568–576.

Specht, D. F. (1992). Enhancements to probabilistic neural networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (Vol. 1, pp. 761–768). Piscataway, NJ: IEEE.

Specht, D. F. (1994). Experience with adaptive probabilistic neural networks and adaptive general regression neural networks. In *Proceedings of the IEEE World Congress on Computational Intelligence* (Vol. 2, pp. 1203–1208). Piscataway, NJ: IEEE.

Traven, H. G. C. (1991). A neural network approach to statistical pattern classification by "semi-parametric" estimation of probability density functions. *IEEE Transactions on Neural Networks, 2*, 366–377.

Tseng, M.-L. (1991). *Integrating neural networks with influence diagrams for multiple sensor diagnostic systems*. Unpublished doctoral dissertation, University of California at Berkley.

Williamson, J. R. (1996). Gaussian ARTMAP: A neural network for fast incremental learning of noisy multi-dimensional maps. *Neural Networks, 9*(5), 881–897.

Williamson, J. R. (1997). A constructive, incremental-learning network for mixture modeling and classification. *Neural Computation, 9*, 1517–1543.

Received May 29, 2006; accepted December 6, 2006.