The probabilistic neural network architecture for high speed classification of remotely sensed imagery

Samir R. Chettri
Hughes-STX at NASA/Goddard Space Flight Center
Greenbelt, MD 20771

Robert F. Cromp
Code 930.1
NASA/Goddard Space Flight Center
Greenbelt, MD 20771

Abstract

In this paper we discuss a neural network architecture (the Probabilistic Neural Net or the PNN) that, to the best of our knowledge, has not previously been applied to remotely sensed data. The PNN is a supervised non-parametric classification algorithm as opposed to the Gaussian maximum likelihood classifier (GMLC).

The PNN works by fitting a Gaussian kernel to each training point. The width of the Gaussian is controlled by a tuning parameter called the window width. If very small widths are used, the method is equivalent to the nearest neighbour method. For large windows, the PNN behaves like the GMLC.

The basic implementation of the PNN requires no training time at all. In this respect it is far better than the commonly used Backpropagation neural network which can be shown to take $O(N^6)$ time for training where N is the dimensionality of the input vector. In addition the PNN can be implemented in a feedforward mode in hardware. The disadvantage of the PNN is that it requires all the training data to be stored. Some solutions to this problem are discussed in the paper.

Finally, we discuss the accuracy of the PNN with respect to the GMLC and the Backpropagation neural network (BPNN). The PNN is shown to be better than GMLC and not as good as the BPNN with regards to classification accuracy.

1 Introduction

High performance computers and sophisticated sensors are responsible for the explosive generation of data for scientific, industrial and commercial uses. NASA faces the same data glut with its current and future missions (including the Earth Observing System and Tropical Rainfall Measuring Mission platforms). At NASA's Goddard Space Flight Center, the Intelligent Data Management group (IDM), within the Information Science and Technology Office (ISTO), has been investigating and developing data and information management systems that can handle the archiving and querying of data produced by Earth and space missions with fast response times. This work has resulted in an Intelligent Information Fusion System (IIFS) for handling and archiving terabyte–sized spatial

databases; and Scheduler/Planner Under Deadlines (SPUDS) to guarantee that response times are maintained [CCS92]. Our current data source of applications is remotely sensed images. However, IIFS and SPUDS are not inherently limited to this data source, having been conceived as general purpose tools.

IDM's research into neural networks has been ongoing since 1989 [CHC89] starting with research into the applicability of the backpropagation paradigm to remotely sensed images. This work has continued, resulting in comparisons of backpropagation with conventional Gaussian Maximum-likelihood classification [CCB92]. Within IIFS/SPUDS, the neural networks act as high speed, low level image classifiers with higher level domain knowledge being provided by decision trees and expert systems. The combination of IIFS and SPUDS provides a scientist with the means to access a database based on image content at varying levels of resolution.

For IDM's purposes, the data glut problem can be divided into two parts. The first part deals with efficient characterization of the data and subsequent archival processes. The second part deals with efficient querying of the data based on platform, content, and spatial and temporal constraints. This paper will deal with the characterization of satellite images and the attendant problems. In particular we discuss the pros and cons of the Probabilistic Neural Network (PNN) with respect to high speed data classification.

In the following text, the PNN is first described. Next, we discuss the advantages of the PNN with respect to backpropagation neural networks (BPNN) and Gaussian Maximum Likelihood Classifiers (GMLC). As a way of addressing the shortcomings of the PNN, we introduce Kohonen's Learning Vector Quantization (LVQ) which helps increase the feedforward speed of the PNN. We then apply the PNN to an image of the Blackhills in South Dakota and discuss the quality of the output. We conclude with a summary of the main results of the paper and reveal our future research plans.

2 The Probabilistic Neural Network (PNN) architecture

The roots of the PNN lie in the histogram evaluation techniques that date back to 1661 ([TT78], [TK76]). Whereas the histogram uses rectangular boxes and quantizes the data axes, the kernel method chooses not to quantize the data axes, instead placing a kernel at each data point in multidimensional space. For an illustration of this process see [Sil86]. In the following discussion, the density estimates are obtained from a set of n observed data vectors $\mathbf{X_1}, \ldots, \mathbf{X_n}$. The actual density is denoted by $f(\mathbf{x})$ and the estimate of the density by $\hat{f}(\mathbf{x})$.

The multivariate estimate of density [Sil86] with kernel K and window width σ is written as

$$\hat{f}(\mathbf{x}) = \frac{1}{n\sigma^d} \sum_{i=1}^{n} K\left\{\frac{1}{\sigma}(\mathbf{x} - \mathbf{X}_i)\right\}. \tag{1}$$

The kernel function satisfies

$$\int_{R^d} K(\mathbf{x}) d\mathbf{x} = 1. \tag{2}$$

Usually the kernel is a unimodal, everywhere positive function. The use of non-positive kernels is still an open research question.

The Gaussian kernel (also discussed in the next section) can be written as

$$K(\mathbf{x}) = (2\pi)^{-d/2} \exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x}). \tag{3}$$

The kernel function and the smoothing parameter are the two choices to be made in the case of density estimates using the kernel method. Research has shown that the choice of kernel does not greatly vary the density estimate [Sil86]. All things being equal, it is then desirable to choose kernels based on their computational properties. We will address this issue in section 2.4.

2.1 Discriminant functions

Given K classes, let $f(\mathbf{X} \mid S_k)$ be the probability density function (pdf) associated with the measurement vector $\mathbf{X}$, given that $\mathbf{X}$ is from class k. Let $P(S_k)$ be the *a priori* probability of class S_k. We can use the **maximum** *a posteriori* (MAP) decision rule to identify the class to which $\mathbf{X}$ belongs. It can be stated as follows ([And72]):

$$\textit{Decide } \mathbf{X} \in S_k \textit{ iff } f(\mathbf{X} \mid S_k)P(S_k) \geq f(\mathbf{X} \mid S_j)P(S_j),\ j = 0, 1, \cdots M-1\ ,$$

where the products $f(\mathbf{X} \mid S_k)P(S_k)$ correspond to discriminant functions, and there are M classes for which discriminant functions are defined. As stated, the MAP rule consists of evaluating the discriminant functions and selecting the maximum as the winner.

Estimating the density function is a key problem in MAP estimation. If the underlying density of each class were known, the problem would be an easy one. In fact, the Gaussian Maximum Likelihood Classifier (GMLC) simplifies the problem of density estimation by *assuming* that $f(\mathbf{X} \mid S_k)$ is a multi-variate normal pdf whose parameters (the mean vector and the variance - covariance matrix) can be determined by samples conditioned on class S_k.

The probabilistic neural net (PNN) was designed by Specht [Spe90] using Parzen's [Par62] kernel function:

$$f(\mathbf{X} \mid S_k) = \frac{1}{(2\pi)^{d/2}\sigma^d} \frac{1}{P_k} \sum_{i=1}^{P_k} \exp\left[-\frac{(\mathbf{X} - \mathbf{W}_{ki})^T(\mathbf{X} - \mathbf{W}_{ki})}{2\sigma^2}\right]. \tag{4}$$

Note that Parzen's kernel is the same as the Gaussian kernel of equation (3). In equation (4) $\mathbf{W}_{ki}$ is the i^{th} training pattern from the $0 \leq k^{th} \leq M-1$ category, P_k is the total number of training patterns in class k, d is the dimension of the training pattern $\mathbf{W}_{ki}$, and σ is a "smoothing According to Specht, a small value of σ caused the density to have modes at the sites of the training samples. Increasing σ causes smoothing of the surface around the modes. In the limiting case, the pdf is Gaussian regardless of the true nature of the underlying distribution. This may seem to be a problem; however, according to Specht, " it is not difficult to find a good value of σ, and ... the misclassification rate does not change dramatically with small changes in σ."

2.2 PNN implementation details

The PNN can be implemented using a feed-forward network. An overview of the PNN is shown in Figure 1. There are four layers. The input layer fans out the input d dimensional vector which has

to be placed in one of M classes. Each node in the input layer is connected to every node in the pattern layer and input vector components are transformed by means of a weight $W_{i,j}$ connecting the i^{th} input node to the j^{th} pattern node. The pattern layer is subdivided into sets of nodes. Each set of nodes does the processing for a particular class. Since there are M classes, there are M sets of pattern nodes. The output of each pattern node set is sent to a node in the summation layer, thus there are M nodes in the summation layer. Finally, the outputs of the summation layer nodes are sent to the decision layer which obtains the maximum output O_k, $k = 0, \ldots, M-1$, and assigns the input vector X to class k.

In equation (4) $-\frac{(\mathbf{X}-\mathbf{W}_{ki})^T(\mathbf{X}-\mathbf{W}_{ki})}{2\sigma^2}$ is exponentiated. This product can be written as $\mathbf{X}^T\mathbf{W} - 1$ if both input and weight vectors are converted to unit vectors, as shown in Figure 2 (a). After the dot product is completed, 1 is subtracted from the total and this is multipied by σ^{-2} after which the exponentiation is performed. At the end of this step one of the terms in the sum of equation (4) has been evaluated.

If the input and weight vectors are not converted to unit vectors, then the architecture of the PNN as shown in Figure 2 (a) can be changed to reflect this. It should be mentioned here that using unit vectors changes the kernel evaluation from a dot product and two vector subtractions to a single dot product and a scalar subtraction. The disadvantage to the dot product method is that magnitude information, that may be useful during the classification process, is lost. On the other hand if our vectors all contain integers, the kernel evaluation process may be done efficiently using integer computations and the dot product method dispensed with entirely.

A summation layer node contains an adder that sums up the outputs of all the pattern nodes in a particular set and then multiplies the output by $\frac{1}{(2\pi)^{d/2}\sigma^d P_k}$ as shown in Figure 2 (b). Thus the summation layer represents the summing process of equation (4).

The decision layer obtains the maximum of the summation layer outputs, and the class to which a given input vector X belongs is finally output.

The PNN is trained by first converting the training exemplars to unit vectors. Next each connection between the input node and a pattern node is assigned a weight which is nothing but an element from the unit training vector. Thus the number of pattern layer nodes corresponds to the number of training vectors and each weight between input and pattern layer nodes corresponds to an element of a training vector. Once training has been done, the network is ready for use in feed-forward mode. The only input parameter from the user in feed-forward mode is σ. A good heuristic method for selection σ is described in [KF72]. In this method the smoothing parameter is given by $\sigma = (d^{-1}\mathrm{tr}\,[\mathbf{C}])^{1/2}N^{-\alpha/d}$. Here, $\mathbf{C}$ is the covariance matrix estimated from the data, d is the dimensions of the data, tr is the trace of a matrix, N is the number of samples and $0 < \alpha < 0.5$. Computing $\mathbf{C}$ would make the PNN training time identical to the Gaussian maximum likelihood estimator, thereby eliminating its main advantage. In [MAC+92], Radial Basis Functions (of which PNN's are a part) are used to reduce the number of hidden nodes by obtaining the covariance $\mathbf{C}$ matrix of samples and also to obtain the widths of the kernel functions. While the RBF approach of [MAC+92] is useful, we have found that σ can take on a reasonably large range of values without seriously affecting accuracy, hence our adherence to the PNN paradigm. A discussion of the results pertaining to our choice of σ is given in section 3.2.

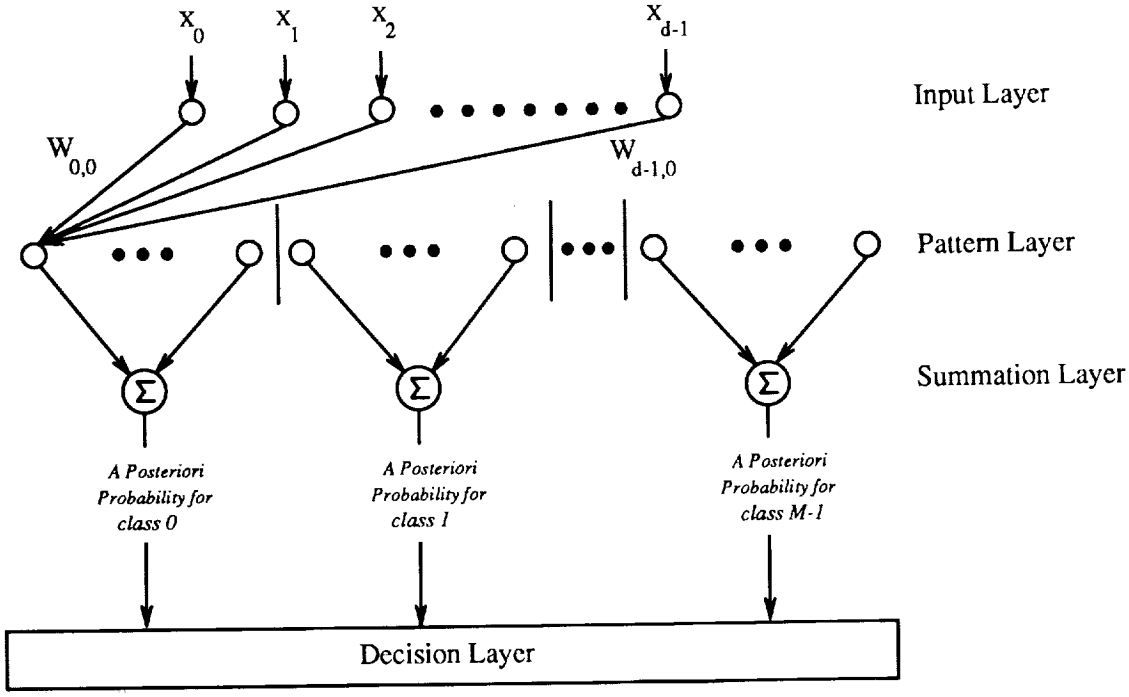

Figure 1: Feed forward implementation of Specht's discriminant analysis method

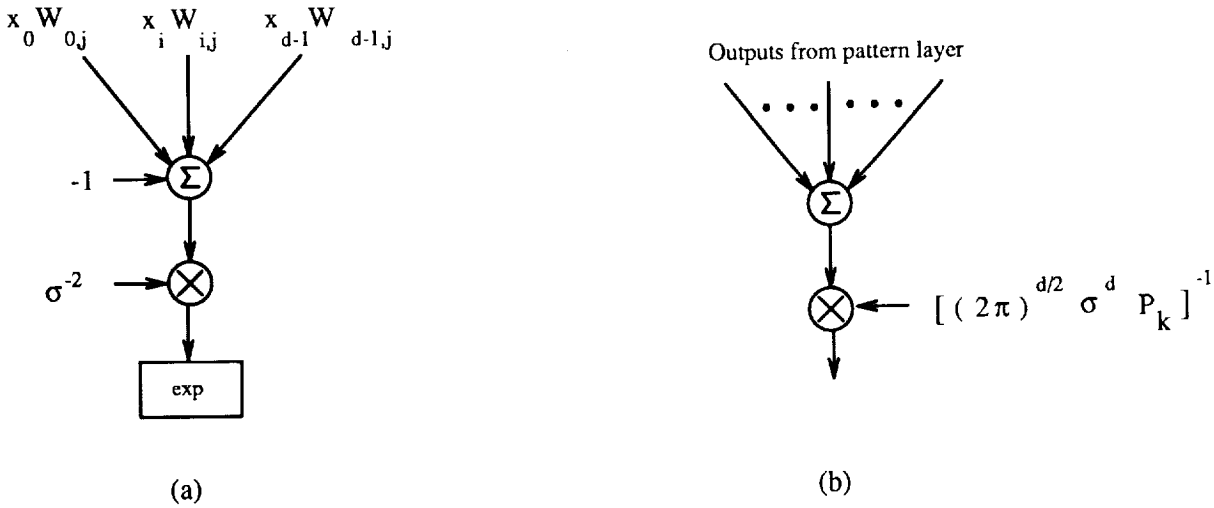

Figure 2: Details of pattern and summation layers

2.3 PNN advantages and disadvantages

In this subsection we discuss the advantages and disadvantages of using PNNs. In particular we focus on training time, retraining time, robustness to weight modification, computational load and memory requirements.

Advantages

1. Training time for the PNN is proportional to the total number of data vectors. In backpropagation the training time is roughly $O(d^6)$ [CCB92] where d is the dimensionality of the input vector. Also, the weights in the resulting backpropagation network do not bear any relationship to the training data and therefore are difficult to interpret. Depending on the flavor of the BPNN chosen, there are several parameters whose values have to be selected through heuristic means. These can affect the accuracy and generalization capability of the net.

   For example, a BPNN of the type described in [HKP91] requires that we randomly initialize the weights. The learning rate and the momentum are two additional parameters to be chosen. Different choices of these free parameters lead to different neural networks with different classification abilities. With the PNN there is only one free parameter σ to be chosen and with the negligible training time, many different nets can quickly be constructed with different values of σ and the best one chosen.

2. Retraining the PNN is easy since the hidden layer can be pruned or enlarged on demand. When a new data vector is received, it can be inserted as a node in the appropriate position in the pattern layer, and the weight connections are made from this node to the input layer. This is an $\mathcal{O}(d)$ process and an attractive feature when compared to BPNN, since the BPNN must be retrained (though not from scratch [SR87]) when new data arrive or when data from the original training set is removed.

3. The PNN is robust to weight removal. Infact, the weight removal and adjustment is the basis for the pruned PNN (PPNN). Our studies (which are also backed up by other recent work [Bur91]) indicate that the number of nodes in the pattern layer can be reduced by considerable amounts and yet give very accurate results. For further discussion on this see section 3.2. In contrast, due to the compact nature of the BPNN, weight deletion/node removal may severely impact classification accuracy.

Disadvantages

1. Since the entire training set is kept (and not an encoded version of it, as in the BPNN), the size of the hidden layer is very large as compared to the BPNN. This would be a shortcoming in computing environments where memory is scarce.

2. Processing speed is slower than BPNN since each input vector has to be evaluated over the entire training set. In a previous publication [CCB92], we have shown that the classification speed for a single input vector for the BPNN is $\mathcal{O}(d^2)$, where d is the length of the input vector. For the PNN, the speed is $\mathcal{O}(M P_k d^2)$, where M is the total number of classes and P_k is the number of exemplars in class S_k. On the surface, this would make the PNN and the BPNN have the same execution time. However, according to Kalayeh and Landgrebe [KL83],

for accurate classification P_k is proportional to d. Hence, in practice the network would have a speed that is $\mathcal{O}(d^3)$.

From our discussion, it is evident that the PNN would be very well suited for exploring dynamic environments. Such environments are commonplace in many scientific investigations of data. One of the goals of IDM's IIFS/SPUDS is the *high speed* classification of data into discipline specific indices for potential users of the system. In its computationally inefficient form, the PNN is incompatible with these goals. In the next section we address this major disadvantage of the PNN.

2.4 Speeding up feed–forward implementations of the PNN

The major disadvantage of the PNN is that *all* the training data are retained in the form of weights. This data can grow extremely large, making feed-forward evaluation of input vectors impossible in real time. One interesting way to prune the hidden layer is described in [Bur91]. In this paper, the author suggests that the data in the hidden layer be pruned using the Learning Vector Quantization algorithm of Kohonen [Koh89]. Unlike [Bur91], where the pruned PNN was applied to simulated bivariate uniform and Gaussian distributions, we apply it to higher dimensional data from real distributions. The LVQ method can be implemented as a feed–forward neural network working in both supervised and unsupervised mode [Sim90]. In the following paragraphs, we give a brief description of the algorithm in its supervised form. For more details the reader is referred to [HKP91].

In LVQ we are given a set of input vectors $\mathbf{X}_i^k$ where $\mathbf{X}_i^k$ represents the i^{th} vector from class S_k, $i = 1, \ldots P_k$ and P_k is the number of training patterns in class S_k. We select M_k vectors per class to represent the P_k vectors where $M_k \ll P_k$. We choose the minimum $M_k < P_k$ such that an acceptable level of accuracy is obtained when the P_k vectors are replaced by the M_k vectors in feedforward mode. This boils down to a trade off between computational efficiency and accuracy and will depend on the user's application. In this network, the initial weight vectors $\mathbf{w}_i^k$ are randomly chosen and a training vector is applied to the neural net input. For each weight vector a set of distances $\left|\mathbf{w}_i^k - \mathbf{X}_i^k\right|$ is calculated and the smallest one (denoted by $\mathbf{w}_{i^*}^k$) is chosen from this set. Next we move this weight closer to the input vector by the following update rules:

$$\Delta \mathbf{w}_{i^*}^k = \begin{cases} +\alpha(t)[\mathbf{X}_i^k - \mathbf{w}_{i^*}^k] & \text{if class is correct} \\ -\alpha(t)[\mathbf{X}_i^k - \mathbf{w}_{i^*}^k] & \text{if class is incorrect} \end{cases} \tag{5}$$

In equation (5) $\alpha(t)$ is the learning rate at time (or iteration level) t. The value of $\alpha(t)$ decreases as the number of iterations in the learning process increases. A common choice is $\alpha(t) = t^{-1}$. The number of iterations is denoted by t_{max}, and its range is $500 \leq t_{max} \leq 10000$. Our practical experience indicates that choosing t_{max} in the range above is sufficient for convergence. In fact, using $t_{max} > 500$ did not lead to great increases in accuracy.

Now the key point in the LVQ pruning of the PNN is that $M_k \ll P_k$, i.e., the number of prototype vectors M_k is much less than the number of vectors P_k in the hidden layer, yet gives an adequate representation of all the P_k vectors in that class. Hence, the time for feedforward classification of input vectors can be decreased, and the memory requirements reduced. We have performed experiments in pruning the PNN so as to improve the feedforward speed. These results

Table 1: Distribution of data, Blackhills, South Dakota

	Training No. of pixels	Entire image No. of pixels	Class name USGS - Level I
0	453	6676	Urban
1	478	42432	Agricultural
2	464	16727	Rangeland
3	482	194868	Forested Land
4	0	0	Water bodies
5	0	0	Wetland
6	368	1441	Barren
7	0	0	Tundra
8	0	0	Perennial snow and ice

are presented in section 3.2, Table 3 and indicate that pruning the PNN is a viable computational scheme.

3 Application of the PNN to remote–sensing

In this section we describe the data on which we tested our PNN, discuss the selection process that we employed for the training and testing data, elaborate upon the training and testing methodology used, and finally, present results for the basic PNN and the LVQ–pruned version of the PNN.

3.1 Description of data set

The data set that was used for training and testing the PNN is called the Blackhills data set, generated by the Landsat 2 multispectral scanner (MSS) (see Figure 3). This data set was previously used to compare backpropagation neural networks with Gaussian maximum likelihood classification in [CCB92]. The spectral bands are 0.5 - 0.6μm (green), 0.6 - 0.7μm (red), 0.8 - 1.1μm (near–infrared). These bands correspond to channels 4 through 7 of the Landsat sensors. There are 262, 144 pixels corresponding to a 512×512 image size, and each pixel represents approximately $79m \times 79m$ on the ground. The image region covers a range of latitudes from 44°15′ to 44°30′ and longitudes from 103°30′ to 103°45′; the images were obtained in September 1973. The ground reference data set was also provided in the form of United States Geological Survey level II land use/land cover data [AHRW76]. Since we were only interested in level I classification, the different classes were conglomerated into the various higher level classes in the hierarchy; the distribution of pixels is shown in column three of Table 1. For example, from Table 1 we know that there are a total of 6676 pixels in the urban class, 453 of which were used for training the PNN.

3.2 Training and testing the PNN

The ground reference data set was viewed on a display device to get an idea of the spatial distribution of the ground reference data pixels. According to [Ric86], a minimum sample size of 60 pixels is necessary for accurate classification. Also, according to [Cam87], a large number of smaller training sites should be used rather than a few large ones. Following these recommendations, we formed training sets from the Blackhills data set. The distribution of training samples is summarized in column two of Table 1.

The PNN classifier is derived from the training group and the error estimate obtained from the test group. This method is known as the "holdout" or H method of estimating errors. We do not use the *leaving-one-out* method of training and testing as described in [WK89] since this method is extremely time consuming and only leads to marginally better accuracy [KC68] in testing for large data sets (which is the case for us).

Results for testing the trained PNN on the image are shown in the contingency table (Table 4). Each entry C_{ij} in the matrix represents the number of times a pixel in class i was put into class j. C_{ii} is the number of correct classifications in class i. In Table 4 the left-hand side set of values represents the classification results of the Unpruned PNN (UPNN) while the pruned PNN (PPNN) results are on the right. The percent correctly classified (PCC) for the UPNN is 0.697, while it is 0.737 for the PPNN. While a 4% difference might seem small, for a 512×512 image this amounts to approximately 10,000 additional pixels being correctly classified. The future generation of Earth orbiting platforms will transmit terabytes to petabytes of data, so small percentage changes in accuracy of classification will lead to large absolute changes in classification accuracy.

A closer look at Table 4) indicates that for forest and urban land cover classes (categories 0 and 3), the PPNN performs better whereas in the other cases, it is not as good. This can be explained on the basis that categories 0 and 3 have less spread in their data vectors (i.e., they are clustered very tightly), hence representing them by a smaller set of vectors leads to no degradation in classification ability. However the samples of the other classes (1, 2, 6) have a larger in-class variability and hence their multidimensional spatial clusters are not compact leading to poorer representation by a smaller set of vectors.

To study the change in the classification ability of the UPNN with σ, we varied σ in increments of 2 from 2 through 12, with the entire training set being used. The results are presented in Table 2. The results indicate that there is some variability in the classification with σ and that as σ grows larger, the UPNN PCC tends toward the Gaussian Maximum Likelihood Estimate with PCC = 0.653 as discussed in [CCB92].

Another test that we performed was to see the variability of the neural net accuracy as we changed the number of hidden nodes in the PPNN. In this test, $\sigma = 4$ since that was the case for which we got maximum PCC in the unpruned PNN. These results are shown in Table 3. We found that as the number of nodes increased there was a general increase in the accuracy of the PPNN. The four node case is an anomaly, since some classes were classified well while other classes were classified extremely poorly (to the extent of having less than 25 pixels put in the rangeland class, i.e., about 0.1% of the total number of pixels).

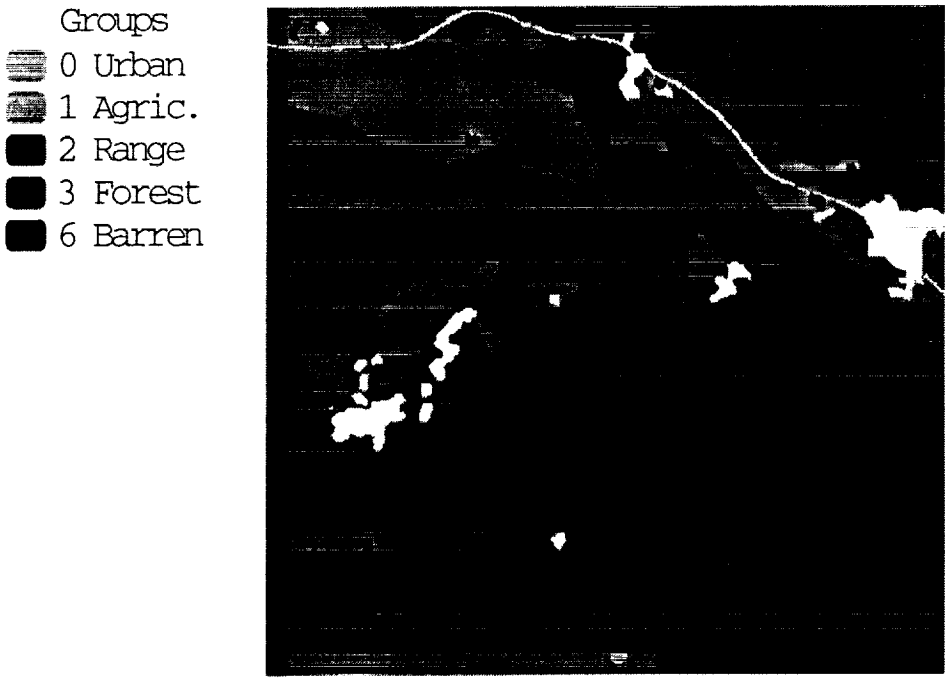

Figure 3: Ground reference data for the Blackhills image

Table 2: Variation of PCC with σ

σ	2	4	6	8	10	12
PCC	0.677	0.697	0.685	0.674	0.670	0.667

Table 3: Variation of PCC with number of nodes/class

nodes/class	4	10	20	40	80
PCC	0.744	0.720	0.721	0.737	0.742

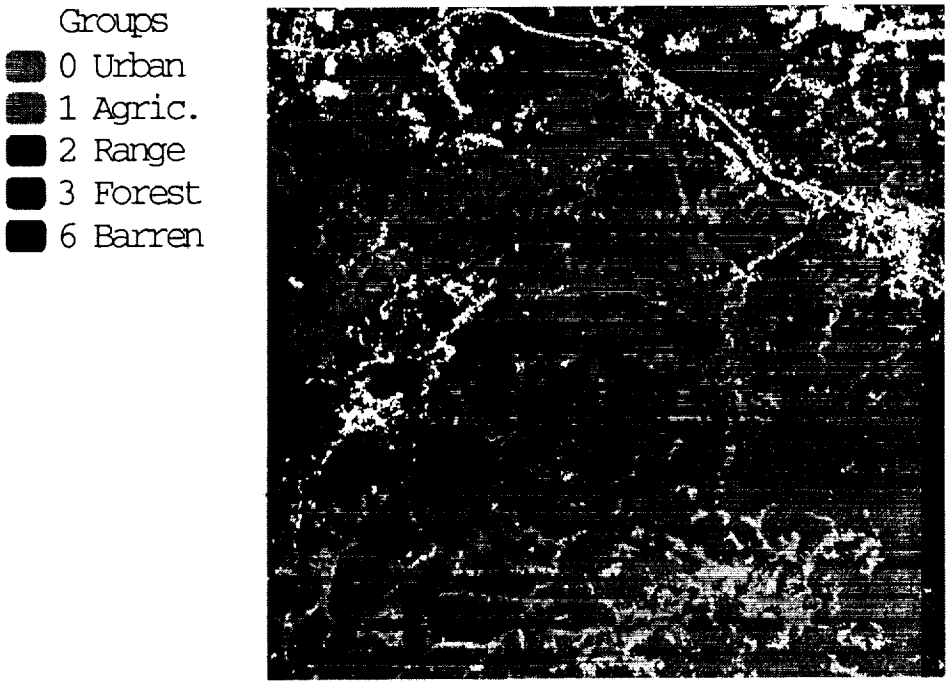

Figure 4: PNN classified Blackhills dataset

Table 4: Contingency table for PNN; Raw PNN on left with $\sigma = 4$ (PCC = 0.697); PNN-LVQ combination on right with $\sigma = 4$ (PCC = 0.737)

	0	**1**	**2**	**3**	**6**	Total pixels	**0**	**1**	**2**	**3**	**6**
0	**0.410**	0.165	0.228	0.153	0.044	6623	**0.509**	0.090	0.147	0.172	0.083
1	0.162	**0.418**	0.378	0.035	0.007	41954	0.215	**0.09**	0.147	0.172	0.083
2	0.110	0.286	**0.532**	0.071	0.002	16263	0.114	0.245	**0.381**	0.129	0.132
3	0.009	0.064	0.139	**0.785**	0.003	194386	0.010	0.070	0.041	**0.861**	0.018
6	0.196	0.118	0.128	0.177	**0.381**	1073	0.257	0.065	0.056	0.534	**0.088**

4 Concluding remarks and future work

In this research, we have described the Probabilistic Neural Network (PNN) and have applied it to remotely sensed imagery. The accuracy obtained by the PNN is better than the Gaussian Maximum Likelihood Classifier (GMLC) and not as good as the Backpropagation Neural Network (BPNN). On the other hand, the training time in the PNN is very small when compared to the other two methods. In addition the network is robust to weight changes and has very small retraining time, making it highly suitable for highly variable and dynamic environments. A modified version of the PNN (LVQ–PNN or PPNN) was discussed and compared with the raw PNN. For the chosen data set, the PPNN performed better than the raw PNN.

Future work includes additional research on the PNN and using the PNN in applications, as described next.

4.1 Planned extensions to the PNN

As has already been discussed, it is sensible to automate the pruning of the PNN. In this process the number of nodes in each class would be allowed to grow or decrease independently of each other such that the PCC per class would be optimized.

In addition, the PNN algorithms described throughout the paper are all readily adaptable for a parallel architecture implementation, most likely on a machine such as the MasPar MP-1, a 16,384 processing element SIMD machine.

4.2 Incorporation of the PNN into a metastrategy

The Intelligent Data Management group has developed a number of automatic characterization algorithms drawn from backpropagation networks, PNN, Adaptive Resonance Theory (ART) networks [CG87], decision trees, Fourier analysis and wavelet theory. Each of these methods has its own unique strengths and weaknesses, and there are cases where one may falter while another excels. We plan to attempt to develop a metastrategy that draws on its knowledge of each of these techniques to produce a hybrid characterization algorithm that performs at least as well as any single one of these components [Fin90].

5 Acknowledgements

A number of people within and without the IDM group have contributed to this work. The authors would like to thank William J. Campbell, George Fekete, Robb Lovell and Nicholas Short, Jr. for their help during this research.

References

[AHRW76] J. R. Anderson, E. E. Hardy, J. T. Roach, and R. E. Witmer. A land use and land cover classification system for use with remote sensor data. Geological Survey Professional Paper 964, United States Government Printing Office, Washington, D.C., 1976.

[And72] H. C. Andrews. *Introduction to Mathematical Techniques in Pattern Recognition.* Wiley–Interscience, New York, 1972.

[Bur91] P. Burrascano. Learning vector quantization for the probabilistic neural network. *IEEE Trans. on Neural Networks*, 2(4):458–461, 1991.

[Cam87] J. B. Campbell. *Introduction to Remote Sensing.* Guilford Press, New York, 1987.

[CCB92] S. R. Chettri, R. F. Cromp, and M. Birmingham. Design of neural networks for classification of remotely sensed imagery. In *1992 Goddard Conference on Space Applications of Artificial Intelligence*, pages 137–149. National Aeronautics and Space Administration, 1992.

[CCS92] R. F. Cromp, W. J. Campbell, and Jr. Short, N. M. An intelligent information fusion system for handling the archiving and querying of terabyte–sized spatial databases. In *International Space Year Conference on Earth and Space Science Information Systems.* American Institute of Physics, 1992.

[CG87] G. Carpenter and S. Grossberg. A massively parallel architecture for a self–organizing ART architecture in a nonstationary world. *Computer Vision, Graphics and Image Processing*, 37:54–115, 1987.

[CHC89] W. J. Campbell, S. E. Hill, and R. F. Cromp. Automatic labeling and characterization of objects using artificial neural networks. *Telematics and Informatics*, 6(3–4):259–271, 1989.

[Fin90] N. V. Findler. *Contributions to a Computer-Based Theory of Strategies.* Springer–Verlag, Berlin, 1990.

[HKP91] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation.* Addison–Wesley, Redwood City, California, 1991.

[KC68] L. Kanal and B. Chandrasekaran. On dimensionality and sample size in statistical pattern recognition. In *Proc. Nat. Electron. Conf.*, pages 2–7, 1968.

[KF72] W. L. G. Koontz and K. Fukunaga. Asymptotic analysis of a nonparametric estimate of a multivariate density function. *IEEE PAMI*, 21:967–974, 1972.

[KL83] H. M. Kalayeh and D. A. Landgrebe. Predicting the required number of training samples. *IEEE Trans. on Patt. Anal. and Mach. Intelligence*, 5:664–667, 1983.

[Koh89] T. Kohonen. *Self-Organization and Associative Memory.* Springer–Verlag, Berlin, 1989.

[MAC+92] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels. On the training of radial basis function classifiers. *Neural Networks*, 5:595–603, 1992.

[Par62] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:308–319, 1962.

[Ric86] J. A. Richards. *Remote Sensing Digital Image Analysis, an Introduction.* Springer–Verlag, Berlin, 1986.

[Sil86] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, 1986.

[Sim90] P. K. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations.* Pergammon Press, New York, 1990.

[Spe90] D. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.

[SR87] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:367–372, 1987.

[TK76] M. E. Tarter and R. A. Kronmal. An introduction to the implementation and theory of nonparametric density estimation. *The American Statistician*, 30:105–112, 1976.

[TT78] R. A. Tapia and J. R. Thompson. *Nonparametric Probability Density Estimation.* Johns Hopkins University Press, Baltimore, 1978.

[WK89] S. M. Weiss and I. Kapouleas. An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Eleventh Int. Joint Conference on Artificial Intelligence*, pages 781–787. American Association of Artificial Intelligence, 1989.