THE JOURNAL OF NAVIGATION (2003), **56**, 291–304. © The Royal Institute of Navigation
DOI: 10.1017/S0373463303002200 Printed in the United Kingdom

Neural Network-Based Geometry Classification for Navigation Satellite Selection

Dah-Jing Jwo and Chien-Cheng Lai

(*National Taiwan Ocean University*)

The neural networks (NN)-based geometry classification for good or acceptable navigation satellite subset selection is presented. The approach is based on classifying the values of satellite Geometry Dilution of Precision (GDOP) utilizing the classification-type NNs. Unlike some of the NNs that approximate the function, such as the back-propagation neural network (BPNN), the NNs here are employed as classifiers. Although BPNN can also be employed as a classifier, it takes a long training time. Two other methods that feature a fast learning speed will be implemented, including Optimal Interpolative (OI) Net and Probabilistic Neural Network (PNN). Simulation results from these three neural networks are presented. The classification performance and computational expense of neural network-based GDOP classification are explored.

KEY WORDS

1. GPS. 2. GDOP. 3. Classification.

1. INTRODUCTION. The Geometric Dilution of Precision (GDOP) is a geometrically determined factor that describes the effect of geometry on the relationship between measurement error and position determination error. It is used to provide an indication of the quality of the solution. Some receiver hardware may be limited to processing limited number of visible satellites. Therefore, it might be necessary to select the satellite subset that offers the best or acceptable solutions. The optimal satellite subset is obtained by minimizing the GDOP value.

Simon and El-Sherief (1995) initially proposed the NN approach to approximate and classify the GDOP. In their research, the back-propagation neural network (BPNN) and Optimal Interpolative (OI) net, were employed for implementing the function approximation and group classification, respectively. The NNs were used to learn the mapping relationships between the entries of a measurement matrix and the eigenvalues of its inverse, and thus generated GDOP. Further work on GDOP approximation has been explored by Jwo and Chin (2002). They proposed three other input-output mapping relationships and performed a comparison on the results based on the four types of network architectures.

In this article, the BPNN, OI Net, and PNN will be employed for classifying the GDOP values. As a classifier, the OI Net features a fast training speed compared with the BPNN at the expense of lower rate of correct classification. The

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

Probabilistic Neural Network (PNN), another of the classification-type NN, is very good in terms of both training speed and correct classification. Performance assessment, including the training speed and correct classification rate for classifying GDOP values using the three types of NNs will be explored and discussed. This paper is organized as follows. In Section 2, preliminary background on GDOP and NNs are provided. The NNs employed in this article, i.e., BPNN, OI Net and PNN, are briefly introduced in Section 3. In Section 4, GDOP classification example using NN is presented and the performance is discussed. The conclusion is given in Section 5.

2. PRELIMINARIES: GDOP AND NEURAL NETWORKS.

The GDOP is defined as

$$\mathrm{GDOP} = \sqrt{\mathrm{trace}(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}} \tag{1}$$

which gives a simple interpretation of how much one unit of measurement error contributes to the derived position solution error for a given situation. It determines the magnification factor of the measurement noise that is translated into the derived solution. The most straightforward approach for obtaining GDOP is to use matrix inversion or the closed-form solution to all combinations and select the minimum one. However, the matrix inversion and closed-form algorithm by computer present computational burden to the navigation computer especially when the number of satellites is large. For further information on the GDOP refer to Kihara and Okada (1984), Stein (1985), Yarlagadda, Ali, Al-Dhahir and Hershey (2000), and Jwo and Chin (2002).

NNs are trainable, dynamic systems that can estimate input-output functions. They have been applied to a wide variety of problems since they are model-free estimator, i.e., without a mathematical model. They have been studied for more than three decades since Rosenblatt first applied single-layer perceptrons to pattern classification learning in the late 1950s. A NN is a network structure consisting of a number of nodes connected through directional links. Each node represents a process unit, and the links between nodes specify the casual relationship between the connected nodes. The learning rule specifies how these parameters should be updated to minimise a prescribed error measure, which is a mathematical expression that measures the discrepancy between the network's actual output and a desired output.

NNs have also been shown to behave well in classification and prediction problems. The BPNN has been a very popular learning algorithm throughout all neural applications. BPNN is very good in functional approximation. Another good feature of BPNN is its pattern classification capability. Nevertheless, it is typical of the heuristic approach and it is associated with long training times with no guarantee of achieving a suitable solution within a reasonable training time. Some of the NNs feature a fast training speed in classification with very good accuracy. The OI and PNN are two examples and will be performed in this article.

In Simon and El-Sherief's paper, the Hecht-Nielson's approach in learning complex, multidimensional functions was employed. The Hecht-Nielson's approach was based on Kolmogorov's Theorem and extended to NNs. The theorem states that any functional $\Re^m \rightarrow \Re^n$ mapping can be exactly represented by a three-layer NN with $(2m+1)$ middle-layer neurons, assuming that the input components are normalized

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

to lie in the range [0,1]:

$$\text{fn: } [0,1]^m \subset \Re^m \rightarrow \Re^n \tag{2}$$

Since $\mathbf{H}^T\mathbf{H}$ is a 4×4 matrix, it has four eigenvalues, λ_i ($i=1 \ldots 4$). It is seen that the four eigenvalues of $(\mathbf{H}^T\mathbf{H})^{-1}$ will be λ_i^{-1}. Based on the fact that the trace of a matrix is equal to the sum of its eigenvalues, Equation (1) can be alternatively represented as

$$\text{GDOP} = (\lambda_1^{-1} + \lambda_2^{-1} + \lambda_3^{-1} + \lambda_4^{-1})^{1/2} \tag{3}$$

where

$$f_1(\vec{\lambda}) = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = \text{trace}(\mathbf{H}^T\mathbf{H}) \tag{4a}$$

$$f_2(\vec{\lambda}) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2 = \text{trace}[(\mathbf{H}^T\mathbf{H})^2] \tag{4b}$$

$$f_3(\vec{\lambda}) = \lambda_1^3 + \lambda_2^3 + \lambda_3^3 + \lambda_4^3 = \text{trace}[(\mathbf{H}^T\mathbf{H})^3] \tag{4c}$$

$$f_4(\vec{\lambda}) = \lambda_1\lambda_2\lambda_3\lambda_4 = \det(\mathbf{H}^T\mathbf{H}) \tag{4d}$$

The $\vec{\lambda}^{-1}$ here is viewed as a functional $\Re^4 \rightarrow \Re^4$ mapping from $\vec{f}$ to $\vec{\lambda}^{-1}$, i.e., $\vec{\lambda}^{-1} = \text{fn}(\vec{f})$. The mapping from $\vec{f}$ to $\vec{\lambda}^{-1}$ is highly nonlinear and cannot be determined analytically but can be precisely approximated or classified by the NN.

3. REVIEW OF NEURAL NETWORKS FOR CLASSIFICATION.

A brief review on the three types of NNs used for classification purpose in this article is provided in this section. The networks include BPNN, OI net and PNN.

3.1. *Back-propagation neural network*. The importance of a NN includes the way a neuron is implemented and how their interconnection/topology are made. The procedure for finding a gradient vector in a network structure is generally referred to as back propagation (BP) since the gradient is calculated in the direction opposite to the flow of the output of each node. The BPNN has been most popular throughout all neural applications. It is a feed-forward, multi-layer perceptron (MLP), supervised learning network. A feed-forward network maps a set of input vectors to a set of output vectors. The basic principle of the BP is to use the gradient steepest descent method to minimize the cost function. The training of neural networks is traditionally based on minimization of the cost function. Suppose a set of training samples is available, the problem can be characterized as choosing the weights (or coupling strengths) of a given network such that the following total squared error is minimized:

$$E = \frac{1}{2}\sum_{k=1}^{n}(d_k - y_k)^2 \tag{5}$$

where n is the number of output variables and d_k and y_k represent the desired and actual output of the kth output neuron, respectively.

The BP topology is made of three layers, one input layer, one hidden layer, and one output layer, as shown in Figure 1. The components in the architecture essentially include: x_i – value of the ith input neuron; h_j – output of the jth hidden neuron; w_{ij} – interconnection weight between the ith input neuron and the jth hidden neuron;

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

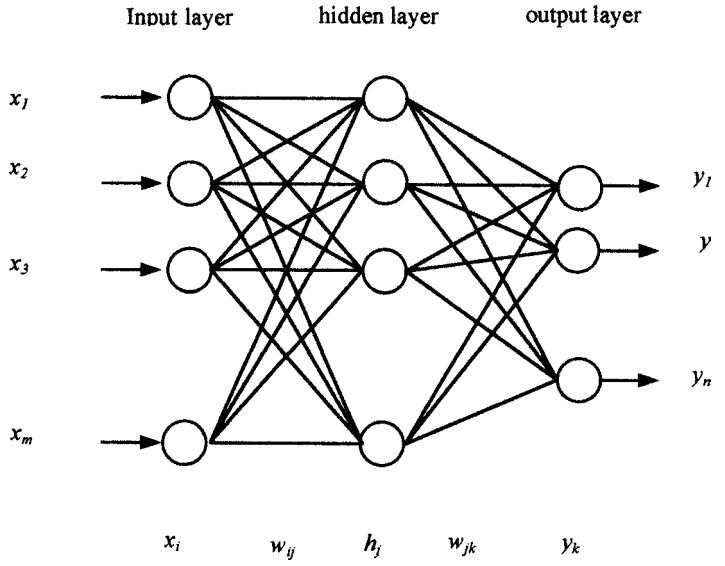


Figure 1. Basic architecture of the BPNN.

w_{jk} – interconnection weight between jth hidden neuron and the kth output neuron. The adjustment of weights is implemented by

$$w_{ij}(n+1)=w_{ij}(n)+\eta\delta_j h_j+\alpha(w_{ij}(n)-w_{ij}(n-1)) \tag{6a}$$

and

$$w_{jk}(n+1)=w_{jk}(n)+\eta\delta_k y_k+\alpha(w_{jk}(n)-w_{jk}(n-1)) \tag{6b}$$

where $(n+1)$, (n), and $(n-1)$ are indices for the next, present, and previous, respectively, and

η: learning rate,

α: momentum,

h_j: a sigmoid function for a hidden neuron, given by

$$h_j=\frac{1}{1+\exp(-v_j)}, \quad \text{in which } v_j=\sum_i w_{ij}x_i-\theta_j$$

y_k: a sigmoid function for an output neuron, given by

$$y_k=\frac{1}{1+\exp(-r_k)}, \quad \text{in which } r_k=\sum_j w_{jk}h_j-\theta_k$$

δ_j: error for a hidden neuron, given by

$$\delta_j=h_j(1-h_j)\sum_k w_{jk}\delta_k$$

δ_k: error for an output neuron, given by

$$\delta_k=y_k(1-y_k)(d_k-y_k)$$

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

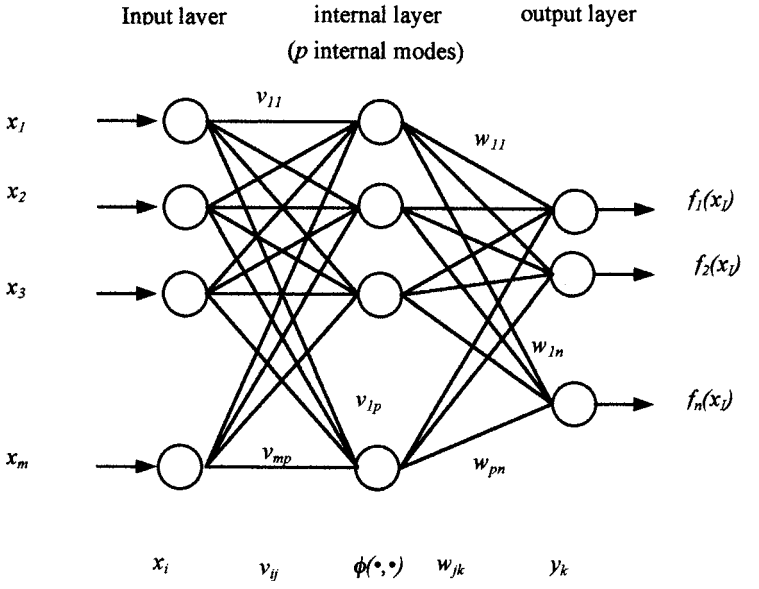


Figure 2. Basic architecture of the OI Net.

The parameters θ_j and θ_k above represent the bias/threshold values of the jth hidden neuron and kth output neuron, respectively. It should be noted from the above that the activation functions of the hidden layer and the output layer are typically sigmoid functions with the form

$$f(u) = \frac{1}{(1+e^{-u})} \tag{7}$$

where $u \in (-\infty, \infty)$, and $f(u) \in (0, 1)$.

The training procedure can be found in Jwo and Chin (2002). For a complete description for the BPNN procedure, see, e.g., Widrow and Lehr (1990), Chester (1993) and Haykin (1999).

3.2. *Optimal Interpolative net*. The Optimal Interpolative (OI) net was initially proposed by deFigueiredo (1990a, 1990b) using a *generalized Fock* (*GF*) *space* formulation. A recursive learning algorithm called RLS-OI was later introduced by Sin and deFigueiredo (1993), which extended the original OI solution to a recursive least squares learning algorithm.

The OI net is also a three-layer feed-forward NN, which has a similar network structure to that of BPNN, as shown in Figure 2. The first layer has m neurons, one for each component of the input. The second layer has p neurons, where p is a number which is chosen during training. The third layer has n neurons, one for each component of the output. v_{ij} is the weight from ith input neuron to the jth middle layer neuron, whereas w_{jk} is the weight from the jth internal node to the kth output node. The weight matrix $\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_p] \in \Re^{m \times p}$ is obtained directly from the components of the exemplars. The vectors $\mathbf{v}_j$ are called prototypes and are chosen from the training set inputs during the learning procedure. The transfer function in the hidden layer is $\varphi(\mathbf{v}_j, \mathbf{x}_i)$, in which $(\cdot,\cdot)$ denotes the dot product of prototype and

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

input vectors. The matrix $\mathbf{W}=[\mathbf{w_1}\ \mathbf{w_2}\ \cdots\ \mathbf{w}_n]\in\Re^{p\times n}$ is the weight matrix to be chosen during training. The kth class of output, i.e., network output from the kth neuron can be represented as

$$f_k(\mathbf{x}_i)=\sum_{j=1}^{p} w_{jk}\cdot\varphi(\mathbf{v}_j,\ \mathbf{x}_i) \tag{8}$$

Suppose a training set with q sets of input-output pairs are given. Each of the q training inputs, $\mathbf{x}_i\in\Re^m$, maps into one of n classes C_j, $j=1\ \ldots\ n$. Let $\mathbf{y}_i\in\Re^n$ be the desired output corresponding to $\mathbf{x}_i$, the output $\mathbf{y}_i$ is then defined as

$$\mathbf{x}_i\in C_j\Rightarrow\mathbf{y}_i=\boldsymbol{\delta}_j \tag{9}$$

where $\boldsymbol{\delta}_j$ is the n-dimensional vector containing all zeros except for the jth element, which is one. Therefore,

$$\mathbf{Y}=[\mathbf{y}_1\ \ \mathbf{y}_2\ \ \cdots\ \ \mathbf{y}_q]=[\delta_{j1}\ \ \cdots\ \ \delta_{jk}\ \ \cdots\ \ \delta_{jq}]\in\Re^{n\times q} \tag{10}$$

where

$$\delta_{jk}=\begin{cases}1 & if\quad j=k\\ 0 & if\quad j\neq k\end{cases} \tag{11}$$

The activation function at each middle layer neuron is given by $\varphi(s)=\exp(s/\rho)$, where ρ is a learning constant chosen by the user.

The matrix $\mathbf{W}$ is determined based on

$$\min_{\mathbf{W}}\|\mathbf{Y}-\mathbf{W}^{\mathrm{T}}\mathbf{G}\| \tag{12}$$

where $\|\cdot\|$ refers to the Euclidean (or Frobenius) norm of a matrix. Solution for $\mathbf{W}$ can be obtained by solving the system of linear equations

$$\mathbf{W}=(\mathbf{G}\mathbf{G}^{\mathrm{T}})^{-1}\mathbf{G}\mathbf{Y}^{\mathrm{T}} \tag{13}$$

The matrix $\mathbf{G}\in\Re^{p\times q}$ is given by

$$\mathbf{G}=\begin{bmatrix}\varphi(\mathbf{v_1},\ \mathbf{x_1}) & \cdots & \varphi(\mathbf{v_1},\ \mathbf{x}_q)\\ \vdots & \ddots & \vdots\\ \varphi(\mathbf{v}_p,\ \mathbf{x_1}) & \cdots & \varphi(\mathbf{v}_p,\ \mathbf{x}_q)\end{bmatrix} \tag{14}$$

A training input is included as a prototype only if it does not induce ill conditioning in $\mathbf{G}\mathbf{G}^{\mathrm{T}}$. This reduces the number of prototypes, and hence limits the number of middle layer neurons in the network. In practice, the learning procedure is presented with q exemplars during training, one at each time. A given exemplar is included in the minimization problem of (12)–(13) only if it cannot be correctly classified by the network which has been trained up to that point. Those exemplars are included in the vectors $\mathbf{z}_i$. Hence $\mathbf{Y}$ and $\mathbf{G}$ in (10) and (14) are replaced with

$$\mathbf{Y}=[\mathbf{y_1}\ \ \mathbf{y_2}\ \ \cdots\ \ \mathbf{y}_l] \tag{15}$$

$$\mathbf{G}=\begin{bmatrix}\varphi(\mathbf{v_1},\ \mathbf{x_1}) & \cdots & \varphi(\mathbf{v_1},\ \mathbf{x}_l)\\ \vdots & \ddots & \vdots\\ \varphi(\mathbf{v}_p,\ \mathbf{x_1}) & \cdots & \varphi(\mathbf{v}_p,\ \mathbf{x}_l)\end{bmatrix} \tag{16}$$

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

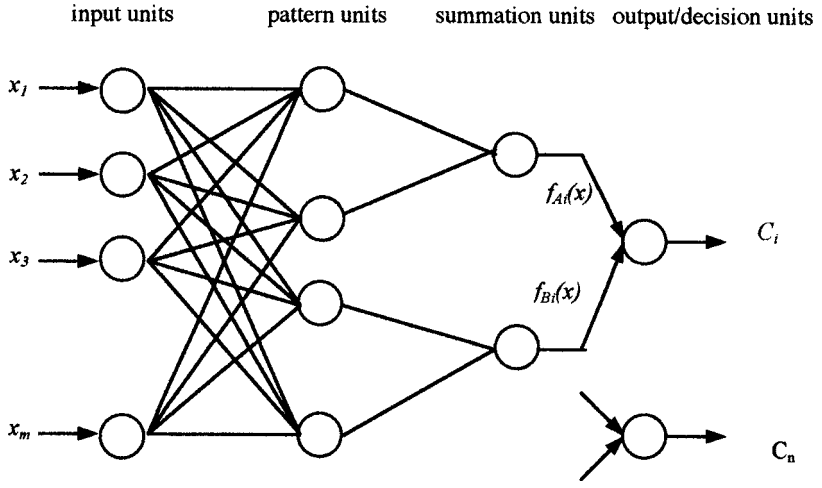


Figure 3. The PNN classifier.

where l is the number of sub-prototypes chosen from the exemplar inputs, and we have the relation $p \leqslant l \leqslant q$. Further discussion on OI Net can be seen in Sin and deFigueiredo (1992), and Dan Simon and Hossny El-Sherief (1995b).

3.3. *Probabilistic neural network* (*PNN*). An alternative to the BPNN and OI net that has been successfully applied in the field of classification is the PNN. It is a four-layer, feed-forward, one-pass training algorithm introduced by Donald Specht (Specht, 1988). The PNN is a universal approximator for smooth class conditional densities. It is able to solve any smooth classification problem given enough data. The PNN is a kind of supervised neural network that has been widely used in the area of pattern recognition, nonlinear mapping, and estimation of probability of class membership and likelihood ratios, See for example, Mao, Tan, and Ser (2000), Tian and Azimi-Sadjadi (2001), Zhang, He, and Mak (2001), Patra, Nayak, Nayak, and Gobbak (2002), and Zaknich (1998).

PNNs are a class of NNs that combine some of the best attributes of statistical pattern recognition and feed-forward NNs. The PNN involves one-pass learning and can be implemented directly in NN architecture. The input variables can be either continuous or binary. The original PNN structure is a direct NN implementation of the Parzen or Parzen-like nonparametric kernel based probability density function (PDF) estimator. It is guaranteed to approach the Bayes' optimal decision surface as the number of training samples increase provided the class PDFs are smooth and continuous. By using sums of spherical Gaussian radial basis functions centred at each training vector to estimate the class PDFs, the PNN is consequently able to make a classification decision in accordance with the Bayes' strategy for decision rules and provide probability and reliability measures for each classification. As just mentioned, the spherical Gaussian function is a Parzen PDF estimator, which can be used to implement the PNN according to:

$$f_i(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\sigma^p} \frac{1}{N_i} \sum_{j=1}^{N_i} \exp\left[-\frac{(\mathbf{x}-\mathbf{w}_{ij})^{\mathrm{T}}(\mathbf{x}-\mathbf{w}_{ij})}{2\sigma^2} \right] \tag{17}$$

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

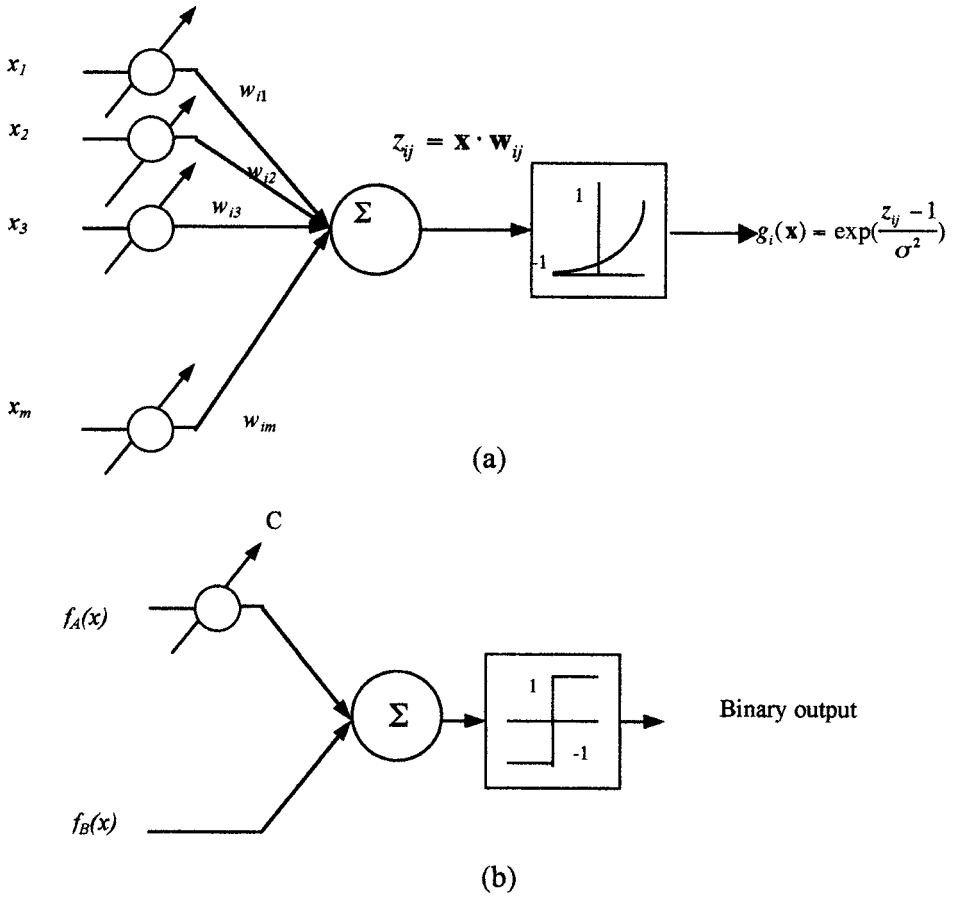


Figure 4. The (a) pattern unit; and (b) output/decision unit of the PNN classifier.

where i and j indicate the class number and pattern number, respectively; d denotes the dimension of the pattern vector $\mathbf{x}$, σ is the smoothing parameter (the standard deviation, to be discussed later), $\mathbf{w}_{ij}$ is the training (or weight) vector from class i (denoted as C_i), and N_i denotes the total number of training vectors in class C_i. The $f_i(\mathbf{x})$ in Equation (17) is used as the basis to the PNN.

The diagram of a PNN classifier, as shown in Figure 3, is a four-layer NN including the input layer, pattern, summation, and decision/output layers. The pattern unit and output unit are shown in more detail in Figure 4. The input layer unit does not perform any computation and simply distributes the input to the neurons in the pattern layer. The pattern units each form a dot product of the input pattern vector $\mathbf{x}$ with a weight vector $\mathbf{w}_{ij}$, $z_{ij} = \mathbf{x} \cdot \mathbf{w}_{ij}$, and then perform a nonlinear operation on z_{ij} before outputting its activation level to the summation unit. Instead of the sigmoid activation function commonly used for BP, the nonlinear operation used in PNN is an exponential activation function

$$g_i(\mathbf{x}) = \exp\left(\frac{z_{ij} - 1}{\sigma^2}\right) \tag{18}$$

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

Table 1. BPNN classification performance for GDOP.

No. of neurons in hidden layer	Misclassification rate after 1,000 epochs (%)	Misclassification rate after 5,000 epochs (%)	Misclassification rate after 10,000 epochs (%)
10	5·42	4·72	3·75
20	3·89	0·83	0·69
30	5·00	1·94	0·83
53	5·14	0·56	0·69

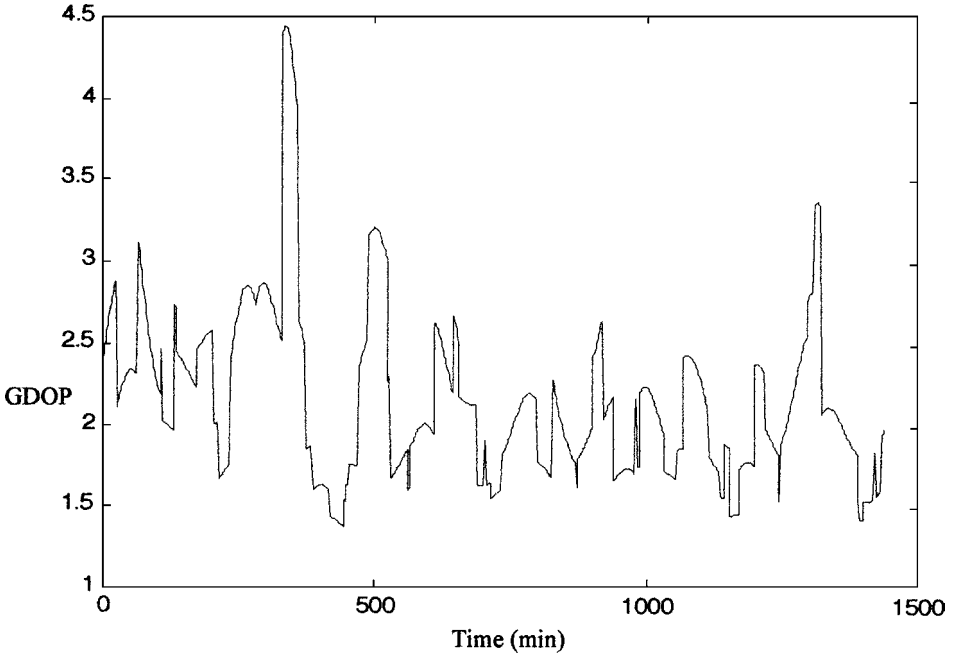


Figure 5. The GDOP plot for simulation in 24-hour duration.

If both $\mathbf{x}$ and $\mathbf{w}_{ij}$ are normalised to unit length, Equation (18) can be written as

$$g_i(\mathbf{x}) = \exp\left[-\frac{(\mathbf{x}-\mathbf{w}_{ij})^{\mathrm{T}}(\mathbf{x}-\mathbf{w}_{ij})}{2\sigma^2}\right] \tag{19}$$

The summation layer neurons compute the maximum likelihood of pattern $\mathbf{x}$ being classified into C_i by summarizing and averaging the output of all neurons that belong to the same class based on Equation (17). If the a priori probabilities for each class are the same, and the losses associated with making an incorrect decision for each class are the same, the decision layer unit classifies the pattern $\mathbf{x}$ in accordance with the Bayes' decision rule based on the output of all the summation layer neurons

$$\hat{C}(\mathbf{x}) = \arg\max\{f_i(\mathbf{x})\}, \quad i = 1, 2, \ldots, n \tag{20}$$

where $\hat{C}(\mathbf{x})$ denotes the estimated class of the pattern $\mathbf{x}$ and n is the total number of classes in the training samples.

The weights are usually 1 or 0 for each hidden unit, a weight of 1 is used for the connection going to the output that the case belongs to, while all other connections are given weights of 0. The only weights that need to be learned are the widths of the

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

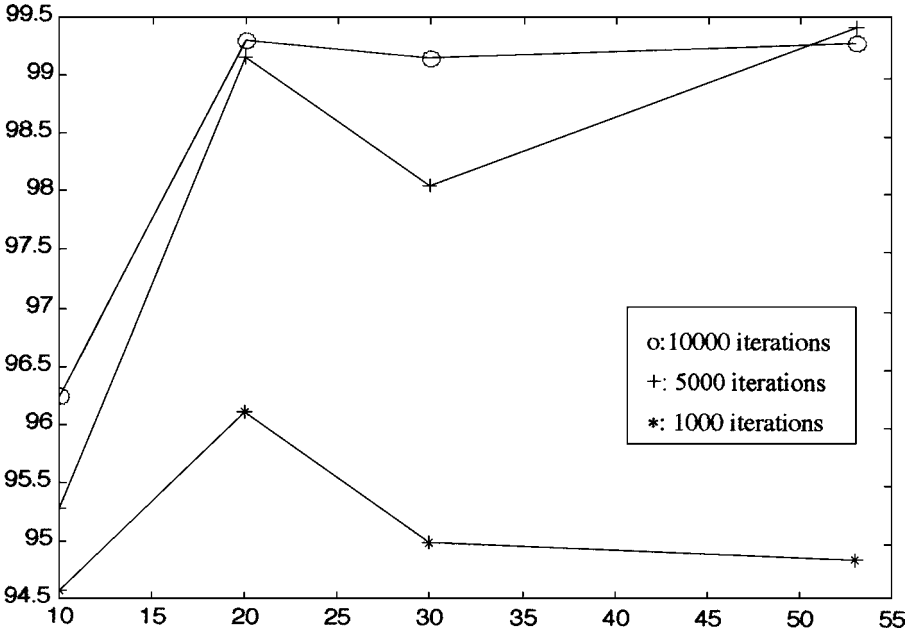


Figure 6. Rate of correct classifications versus number of hidden layer neurons by BPNN.

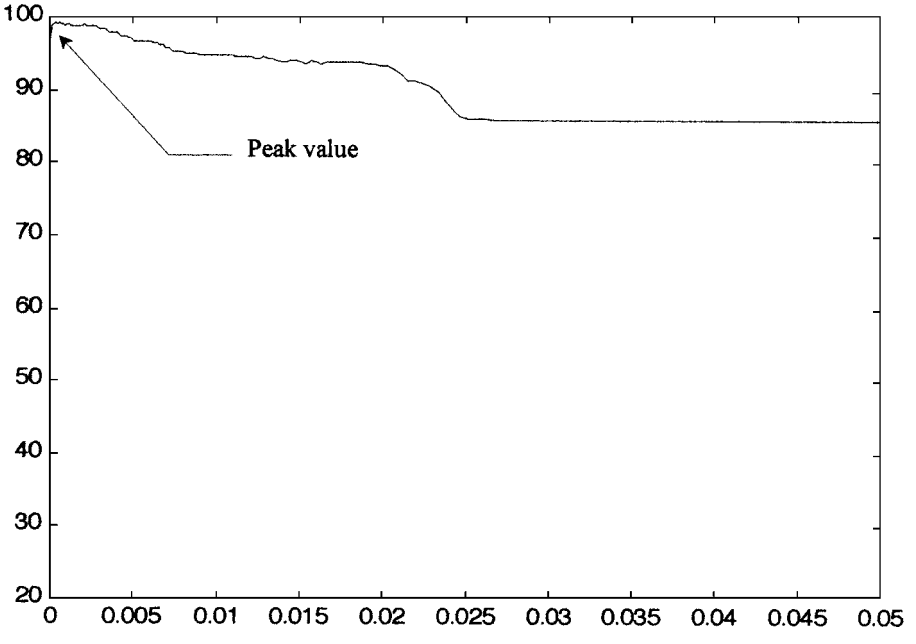


Figure 7. Rate of correct classifications versus smoothing parameter using PNN.

units. One can adjust these weights for the prior probabilities of each class. These widths are the smoothing parameters σ, which as the common standard deviation for all the Gaussians, is the only adjustment made for optimising the network, which is usually chosen by cross validation. The decision boundary of the PNN varies

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

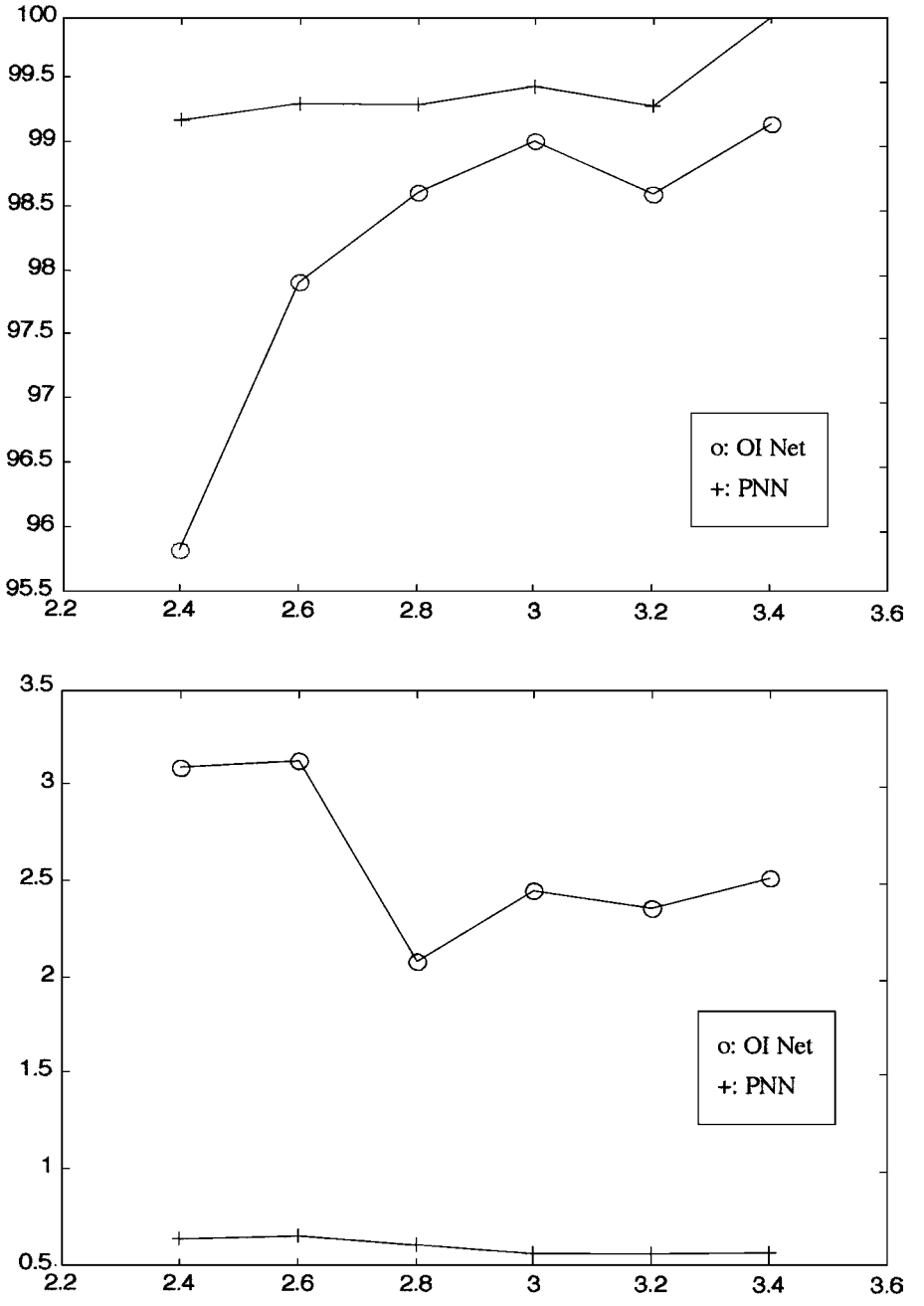


Figure 8. (a) Rate of correct classifications; and (b) training time, versus GDOP threshold by OI Net and PNN.

continuously approaching a matched filter as σ approaches infinity. When σ approaches zero, the function becomes a very nonlinear boundary representing the nearest neighbour classifier. In practical problems it is not difficult to find a good value of smoothing factor by trial and error.

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

Table 2. Classification performance of OI Net and PNN (optimal case).

	OI Net	PNN
Misclassification rate	1∼5%	0∼1%
Training time	2∼3 secs	<1 sec

The main drawback of PNN is that, like kernel methods in general, it suffers badly from the curse of dimensionality. Although the training scheme for PNN is very simple and fast, one major drawback is that potentially a very large network will be formed since every training pattern needs to be stored and used to classify vectors. Hence the useful features of being simple and fast come at the expense of larger memory requirements and slower execution speed for prediction of unknown patterns compared to conventional NNs. Nevertheless, this will not be a severe disadvantage if the PNN is implemented in a parallel hardware structure where memory is relatively inexpensive.

4. GDOP CLASSIFICATION USING NN. Some of the GPS receivers may not be able to process all the visible satellites due to the limited number of channels. To choose a subset of satellites for navigation use, the GDOP classifier will be employed for selecting one of the acceptable subsets (e.g., one with sufficiently small GDOP value).

In this article, the NN classifiers have two types of output based on the GDOP values. A threshold value is defined for certain specific requirement. When the GDOP is smaller than the threshold, an output vector [1 0] is defined, whereas when the GDOP is larger than the threshold, an output vector [0 1] is defined. The input $\vec{f}$ has four input variables. The input variables f_i are normalized to the range [0,1]. The output will have only one variable: 1 or 0. Extension to classification into three types of output can simply be done by defining two thresholds and specifying three output vector: [1 0 0], [0 1 0], and [0 0 1], which stand for the small, medium, and large GDOP, respectively. Similarly, classification into more types of output is feasible based on the same philosophy.

Computer simulation examples are provided to illustrate the classification performance by the NNs. Simulation was conducted using a Pentium III 733 MHz computer. The computer codes were constructed using the Matlab® 6.0 version software. The receiver is simulated to be located at the top of the building of this Department, which has an approximate position of North 25·15 degrees, East 121·78 degrees, at an altitude of 62 m ($[-3042329{\cdot}2 \;\; 4911080{\cdot}2 \;\; 2694074{\cdot}3]^{\mathrm{T}}$ m in the WGS-84 ECEF coordinate) with an mask angle of 5 degrees. A 24-hour duration was simulated. The GDOP was computed every odd minute, and then collected in two files every even minute, one for training the other for testing purposes. Consequently, there are 720 patterns for both the training and testing procedures. Figure 5 provides the GDOP plot for 24-hour duration in the simulation. The results are presented as follows.

4.1. *BPNN classification performance*. When the GDOP threshold is set at 2·5, 152 samples among 720 are smaller than the threshold. The learning rate and momentum for the BPNN are 0·5 and 0·6, respectively. Three different training iterations, 1000, 5000, and 10 000, are selected. Improvement by increasing the number

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

of neurons in the hidden layer is not guaranteed/observed when more than 20 neurons are used. The convergence speed becomes very slow after 3000 epochs of training. The misclassification rates for different number of training epochs are summarized in Table 1 and Figure 6 provides a plot of the correct classification rates for comparison. One point of note for the BPNN classification is that there is a very long training time. Even for the simplest case of 10 hidden layer neurons with 1000 learning iterations, approximately one hour is required for completing the training.

4.2. *OI and PNN classification performance*. The parameters used in the OI Net include a fitting parameter $\rho = 0{\cdot}1$, an ill-conditioning threshold $\gamma_1 = 1e-8$, and an error reduction threshold $\gamma_2 = 1e-3$. See Simon and El-Sherief (1995a) and Sin and deFigueiredo (1992) for further discussion on selecting these parameters. Different GDOP thresholds have been utilized for comparison. The correct classification rate versus the smoothing parameter σ is shown in Figure 7. It shows that the classification rates have better accuracy when σ is very small, and reaches the peak value at approximately $\sigma = 5e-4$. Figure 8 provides performance evaluation for OI Net and PNN. It is seen that the PNN have better performance than OI Net in terms of correct classification rate and training time. The results on classification performance of OI Net and PNN are summarized in Table 2. The accurate classification rates of OI Net are in the range $95 \sim 99\%$; the training times are about $2 \sim 3$ seconds. The accurate classification rates of PNN are in the range $99 \sim 100\%$; the training times in all cases are less than 1 second.

5. CONCLUSION. The NN-based GDOP classification has been successfully conducted. Three types of NNs have been performed and their performances have been explored and discussed. Although it is recognized that the BPNN has been most popular throughout all neural applications, it is also well known to have some drawbacks, particularly slow learning. Use of some classification-type of NNs to classify GDOP values does not have this problem. OI Net and especially PNN have shown very good performance. The PNN has the best rate of correct classification, which almost reaches 100% with a learning time of less than 1 second.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the National Science Council of the Republic of China under Grant NSC 90-2212-E-019-016.

REFERENCES

Simon, D. and El-Sherief, H. (1995a). Navigation satellite selection using neural networks, *Neurocomputing* **7**, pp. 247–258.

Simon, D. and El-Sherief, H. (1995b). Fault-tolerant training for optimal interpolative nets, *IEEE Trans. Neural Networks* **6** (3), pp. 1531–1535.

Jwo, D.-J. and Chin, K.-P. (2002). Applying back-propagation neural networks to GDOP approximation, *This Journal* **54** (2), pp. 97–108.

Kihara, M. and Okada, T. (1984). A satellite selection method and accuracy for the global positioning system, *NAVIGATION: Journal of the Institute of Navigation* **31** (1), pp. 8–20.

Stein, B. A. (1985). Satellite Selection Criteria During Altimeter Aiding of GPS, *NAVIGATION: Journal of the Institute of Navigation* **32** (2), pp. 149–157.

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press

Yarlagadda, R., Ali, I., Al-Dhahir, N. and Hershey, J. (2000). GPS GDOP metric, *IEE Proc.-Radar, Sonar Navig*. **147** (5), pp. 259–264.

Sin, S.-K. and deFigueiredo, R. J. P. (1992), An evolution-oriented learning algorithm for the optimal interpolative net, *IEEE Trans. Neural Networks* **3** (26), pp. 315–323.

Widrow, B. and Lehr, M. A. (1990). 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation, *Proc. of the IEEE* **78** (9), pp. 1415–1442.

Chester, M. (1993). Neural networks: a tutorial, Prentice-Hall.

Haykin, S. (1999). Neural networks: a comprehensive foundation, Prentice-Hall.

Zaknich, A. (1998). Internet: http://www.maths.uwa.edu.au/∼rkealley/ann_all/.

Specht, D. F. (1988). Probabilistic neural networks for classification, mapping, or associative memory, 1988, *IEEE International Conference on Neural Networks* **1**, pp. 525–532.

Mao, K. Z., Tan, K. C., and Ser, W. (2000), Probabilistic neural-network structure determination for pattern classification, *IEEE Trans. Neural Networks* **11** (4), pp. 1009–1016.

Tian, B.; Azimi-Sadjadi, M. R. (2001), Comparison of two different PNN training approaches for satellite cloud data classification, *IEEE Trans. Neural Networks* **12** (1), pp. 164–168.

Zhang, W. D., He, M. X., and Mak, M. W. (2001), Cloud detection using probabilistic neural networks, IEEE 2001 International Geoscience and Remote Sensing Symposium, IGARSS '01 **5**, pp. 2373–2375.

Patra, P. K., Nayak, M., Nayak, S. K. and Gobbak, N. K. (2002). Probabilistic neural network for pattern classification, Proceedings of the 2002 International Joint Conference on Neural Networks, IJCNN '02 **2**, pp. 1200–1205.

https://doi.org/10.1017/S0373463303002200 Published online by Cambridge University Press