GPS Solut (2007) 11: 51–60
DOI 10.1007/s10291-006-0030-z

ORIGINAL ARTICLE

Dah-Jing Jwo
Chien-Cheng Lai

Neural network-based GPS GDOP approximation and classification

Received: 22 September 2005
Accepted: 10 April 2006
Published online: 16 May 2006
© Springer-Verlag 2006

D.-J. Jwo (✉)
Department of Communications and Guidance Engineering, National Taiwan Ocean University, 2 Pei-Ning Road, 202 Keelung, Taiwan, ROC
E-mail: djjwo@mail.ntou.edu.tw
Tel.: +886-2-24622192
Fax: +886-2-24633492

C.-C. Lai
Department of 3G Application, Wintec Wireless Electronics, 716 Jung Jeng Road, Chung Ho, 235 Taipei, Taiwan, ROC
E-mail: al@mail.wintec.com.tw
Tel.: +886-2-82273601
Fax: +886-2-82273548

Abstract In this paper, the neural network (NN)-based navigation satellite subset selection is presented. The approach is based on approximation or classification of the satellite geometry dilution of precision (GDOP) factors utilizing the NN approach. Without matrix inversion required, the NN-based approach is capable of evaluating all subsets of satellites and hence reduces the computational burden. This would enable the use of a high-integrity navigation solution without the delay required for many matrix inversions. For overcoming the problem of slow learning in the BPNN, three other NNs that feature very fast learning speed, including the optimal interpolative (OI) Net, probabilistic neural network (PNN) and general regression neural network (GRNN), are employed. The network performance and computational expense on NN-based GDOP approximation and classification are explored. All the networks are able to provide sufficiently good accuracy, given enough time (for BPNN) or enough training data (for the other three networks).

Keywords GPS · GDOP · Neural networks · Approximation · Classification

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 error. It is used to provide an indication of the quality of the solution. Some of the GPS receivers may not be able to process all visible satellites due to limited number of channels. Consequently, it is sometimes necessary to select the satellite subset that offers the optimal or acceptable solutions. The optimal satellite subset is sometimes obtained by minimizing the GDOP factor.

The most straightforward approach for obtaining GDOP is to use matrix inversion to all combinations and select the minimum one. However, the matrix inversion by computer presents a computational burden to the navigation computer. For the case of processing four satellite signals, it has been shown that GDOP is approximately inversely proportional to the volume of the tetrahedron formed by four satellites (Kihara and Okada 1984; Stein 1985). Therefore, it is optimum to select satellite such that the volume is as large as possible, which is sometimes called the maximum volume method. However, it is not universal acceptable since it does not guarantee optimum selection of satellites.

The neural network (NN) approach provides a promising and very realistic computational alternative. The application of NN approach for navigation solution processing has not been widely explored yet in the GPS community. Simon and El-Sherief (1995a) initially proposed the NN approach to approximate and classify the GDOP factors for the benefit of computational efficiency, where it could be seen that a total of 160 floating point operations (add, subtract, multiply, or divide) are required for each 4×4 matrix inversion in order to determine GDOP using the LU-decomposition method. There are basically two types of strategy employed for justifying satellite geometry based on GDOP, i.e., approximation and classification. Differing from the GDOP approximator for computing the value to select the optimal subsets of satellites, the GDOP classifier is employed for selecting one of the acceptable subsets (e.g., ones with GDOP factor sufficiently small) of satellites for navigation use. In Simon and El-Sherief's research, the back-propagation neural network (BPNN) and optimal interpolative (OI) Net, respectively, were employed for performing the function approximation and group classification, respectively. The NNs were used to learn the functional relationships between the entries of a measurement matrix and the eigenvalues of its inverses, and thus generated GDOP. Extension work on BPNN based GDOP approximation has been explored by Jwo and Chin (2002), where they proposed three other input–output mapping relationships and evaluated the results based on four types of mapping topology.

Although the BPNN has been the most popular learning algorithm throughout all neural network applications and can be employed as an approximator as well as a classifier, it usually requires a very long training time. To overcome the problem of long training time, three other networks are employed. They are the OI Net, probabilistic neural network (PNN) and general regression neural network (GRNN). For function approximation, the GRNN is employed in addition to the BPNN; while for group classification, the OI Net, PNN and GRNN are employed in addition to the BPNN. This paper is organized as follows. In the section "Preliminaries: GDOP and neural networks", the preliminary background on GDOP and neural networks is briefly reviewed. The NNs employed in this paper, i.e., BPNN, OI Net, PNN and GRNN, are introduced in the section "Neural networks for approximation and classification". The section "GDOP approximation and classification using neural networks" provides the mapping topology for performing the GDOP function classification and group approximation. In the section "Simulation and discussion", simulation examples and discussion of results using NNs are presented. Conclusions are given in the section "Conclusions".

Preliminaries: GDOP and neural networks

The least squares solution to the linearized GPS pseudorange equation, $\mathbf{z} = \mathbf{Hx} + \mathbf{v}$, is given by (Yalagadda et al. 2000)

$$\hat{\mathbf{x}} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{z}, \tag{1}$$

where the dimension of the geometry matrix $\mathbf{H}$ is $n \times 4$ with $n \geq 4$. The quality of navigation solution for the linearized pseudorange equation is obtained by taking the difference between the estimated and true positions:

$$\tilde{\mathbf{x}} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{v}, \tag{2}$$

where $\mathbf{v}$ has zero mean, and so does $\tilde{\mathbf{x}}$. The covariance between the errors in the components of the estimated position is:

$$E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{\mathrm{T}}\} = (\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}\mathbf{H}^{\mathrm{T}}E\{\mathbf{v}\mathbf{v}^{\mathrm{T}}\}\mathbf{H}(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1},$$

where $E\{\cdot\}$ is the expected value operator. If all components of $\mathbf{v}$ are pairwise uncorrelated and have variance σ^2, then $E\{\mathbf{v}\mathbf{v}^{\mathrm{T}}\} = \sigma^2\mathbf{I}$, and consequently:

$$E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{\mathrm{T}}\} = \sigma^2(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}. \tag{3}$$

The GDOP factor is defined as

$$\mathrm{GDOP} = \sqrt{\mathrm{trace}(\mathbf{H}^{\mathrm{T}}\mathbf{H})^{-1}} = \sqrt{\frac{\mathrm{trace}[\mathrm{adj}(\mathbf{H}^{\mathrm{T}}\mathbf{H})]}{\det(\mathbf{H}^{\mathrm{T}}\mathbf{H})}}. \tag{4}$$

It is seen that the GDOP factor 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.

NNs are trainable, dynamic systems that can estimate input–output functions and have been applied to a wide variety of problems since they are model-free estimators, 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. The importance of a NN includes the way a neuron is implemented and how their interconnection/topology is made.

Neural networks for approximation and classification

Brief review on four types of NNs, including BPNN, OI net, PNN and GRNN, is provided.

The Back-propagation Neural Network (BPNN)

The BPNN is a feed-forward, multi-layer perceptron (MLP), supervised learning network, which maps a set of input vectors to a set of output vectors. The procedure of finding a gradient vector in a network structure is generally referred to as back propagation (BP). The basic principle of the BP is to use the gradient steepest descent method to minimize the cost function:

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

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

The BP topology is made of three layers, one input layer, one hidden layer, and one output layer. The BPNN essentially includes: 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; w_{jk}-interconnection weight between the 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: sigmoid function for a hidden neuron:

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

y_k: sigmoid function for an output neuron:

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

$\boldsymbol{\delta}_j$: error for a hidden neuron:

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

$\boldsymbol{\delta}_k$: error for an output neuron:

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

The parameters θ_j and θ_k represent the bias/threshold values of the jth hidden neuron and kth output neuron, respectively. The activation functions of the hidden layer cand the output layer are typically sigmoid functions:

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

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 of the topic, see Widrow and Lehr (1990), Chester (1993), and Haykin (1999).

The optimal interpolative (OI) net

The OI Net was proposed by DeFigueiredo (1990) using a *generalized Fock space* formulation (Defigueiredo 1983). A recursive least squares learning algorithm called RLS-OI was subsequently introduced by Sin and DeFigueiredo (1992).

Shown in Fig. 1, the OI net also belongs to a three-layer feed-forward NN, which has a similar network structure to that of BPNN. The first layer has m neurons, one for each component of the input; the second layer has p (to be determined during training) neurons; the third layer has n neurons, one for each component of the output. v_{ij} is the weight from ith input node to the jth internal node, whereas w_{jk} is the weight from the jth internal node to the kth output node. The weigh 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 $\mathbf{v}^j$, called prototypes, are chosen from the training set inputs during the learning procedure. The transfer function in the hidden layer is $\varphi(\mathbf{v}^j, \mathbf{x}^i)$, where $(\cdot, \cdot)$ denotes the dot product. 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 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, $\mathbf{x}^i \in \Re^m$ $(i = 1, 2, \ldots, q)$, are given and each of them maps into one of n classes $C_k(k = 1, 2, \ldots, n)$. 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

$$\mathbf{x}^i \in C_k \Rightarrow \mathbf{y}^i = [y_1^i \quad y_2^i \quad \cdots \quad y_n^i]^{\mathrm{T}} = \boldsymbol{\delta}_j, \tag{9}$$

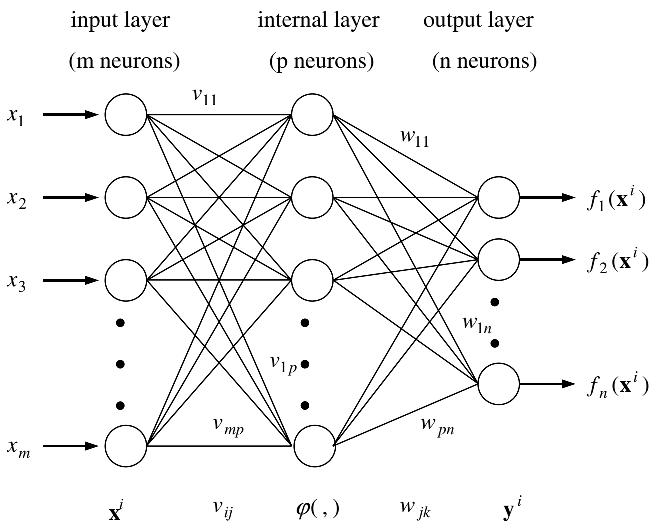


Fig. 1 Basic architecture of the OI Net

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

$$\mathbf{Y} = [\mathbf{y}^1 \quad \mathbf{y}^2 \quad \cdots \quad \mathbf{y}^q] = \begin{bmatrix} y_1^1 & y_1^2 & \cdots & y_1^q \\ y_2^1 & y_2^2 & \cdots & y_2^q \\ \vdots & \vdots & \cdots & \vdots \\ y_n^1 & y_n^2 & \cdots & y_n^q \end{bmatrix} \in \Re^{n\times q}, \tag{10}$$

where

$$y_j^i = \delta_{jk} = \begin{cases} 1 & if \quad j = k \\ 0 & if \quad j \neq k \end{cases}, \quad j = 1, 2, \ldots, n \tag{11}$$

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

Given the matrix $\mathbf{G} \in \Re^{p\,\times\,q}$

$$\mathbf{G} = \begin{bmatrix} \varphi(\mathbf{v}^1, \mathbf{x}^1) & \cdots & \varphi(\mathbf{v}^1, \mathbf{x}^q) \\ \vdots & \ddots & \vdots \\ \varphi(\mathbf{v}^p, \mathbf{x}^1) & \cdots & \varphi(\mathbf{v}^p, \mathbf{x}^q) \end{bmatrix} \tag{12}$$

the matrix $\mathbf{W}$ is determined based on the minimization principle

$$\min_{\mathbf{W}} \left\| \mathbf{Y} - \mathbf{W}^{\mathrm{T}}\mathbf{G} \right\|_2, \tag{13}$$

where $||\cdot||$ refers to the Euclidean norm of a matrix. Solution for $\mathbf{W}$ is

$$\mathbf{W} = (\mathbf{G}\mathbf{G}^{\mathrm{T}})^{-1}\mathbf{G}\mathbf{Y}^{\mathrm{T}} \in R^{p\times n}. \tag{14}$$

A training input is included as a prototype only if it does not induce ill conditioning in $\mathbf{GG}^{\mathrm{T}}$. This reduces the number of prototypes, and hence limits the number of middle layer neurons. The learning procedure is presented with q exemplars during training, one at each time. A given exemplar is included in the minimization problem (Eqs. 13, 14) 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 $\mathbf{z}^i$ (sub-prototypes). Hence, $\mathbf{Y}$ and $\mathbf{G}$ in Eqs. 10 and 12 are replaced with

$$\mathbf{Y} = [\mathbf{y}^1 \quad \mathbf{y}^2 \quad \cdots \quad \mathbf{y}^l], \tag{15}$$

$$\mathbf{G} = \begin{bmatrix} \varphi(\mathbf{v}^1, \mathbf{z}^1) & \cdots & \varphi(\mathbf{v}^1, \mathbf{z}^l) \\ \vdots & \ddots & \vdots \\ \varphi(\mathbf{v}^p, \mathbf{z}^1) & \cdots & \varphi(\mathbf{v}^p, \mathbf{z}^l) \end{bmatrix}, \tag{16}$$

where $p \leq l \leq q$ is the number of sub-prototypes chosen from the exemplar inputs. Further discussion on OI Net can be seen in Defigueiredo (1990), Sin and Defigueiredo (1992), and Simon and El-Sherief (1995b).

The probabilistic neural network (PNN)

Both PNN and GRNN are variants of radial basis function network. Introduced by Donald Specht (1988) as a four-layer, feed-forward, one-pass training algorithm, the PNN is a supervised neural network that can solve any smooth classification problem given enough data.

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 functions centered at each training vector to estimate the class PDFs, the PNN is able to make a classification decision in accordance with the Bayes' strategy for decision rules and provide probability and reliability measures for each classification. The spherical Gaussian radial basis function can be used to implement the PNN according to

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

where i and j indicate the class number and pattern number, respectively; d is the dimension of the pattern vector $\mathbf{x}$; σ is the smoothing parameter; $\mathbf{w}_{ij}$ is the training (or weight) vector from class i (C_i);M_i denotes the total number of training vectors in class C_i.

As shown in Fig. 2, the four layers of PNN classifier include the input, pattern, summation, output/decision layers. The input units do not perform any computation and simply distributes the input to the neurons in the pattern layer. The pattern unit and output unit are shown in more detail in Fig. 3. The pattern units $z_{ij} = \mathbf{x}\cdot\mathbf{w}_{ij}$, and a nonlinear operation using an exponential activation function on z_{ij} is performed

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

before outputting its activation level to the summation unit. If both $\mathbf{x}$ and $\mathbf{w}_{ij}$ are normalized to unit length, Eq. 18 becomes

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

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 Eq. 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 vector $\mathbf{x}$ 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 vector $\mathbf{x}$ 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 to be learned are the widths of the units, which are the smoothing parameters σ (the standard deviation for the Gaussians), which is the only adjustment made for optimizing the network and is usually chosen by cross validation.

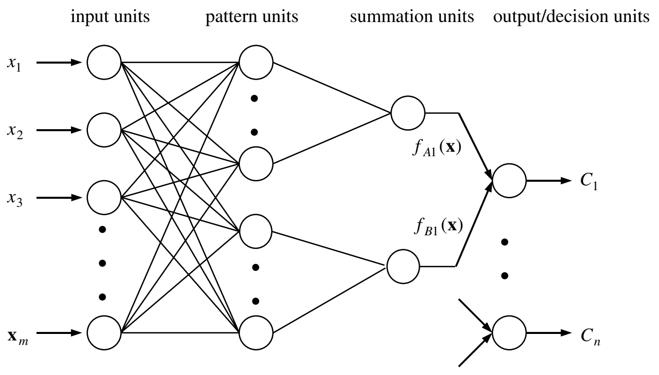
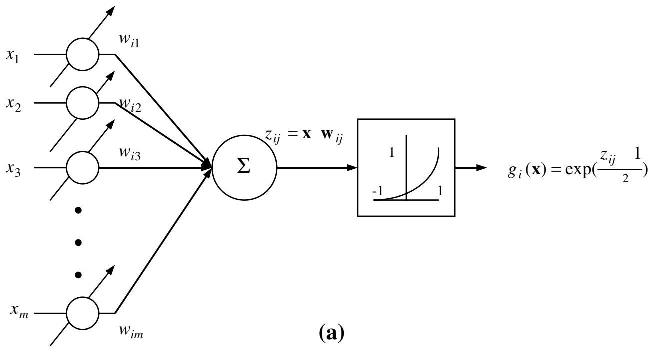


Fig. 2 The PNN classifier


x_1
x_2
x_3
x_m
w_{i1}
w_{i2}
w_{i3}
w_{im}
Σ
$z_{ij} = \mathbf{x}\ \mathbf{w}_{ij}$
1
-1
1
$g_i(\mathbf{x}) = \exp(\frac{z_{ij}\ \ 1}{2})$
(a)

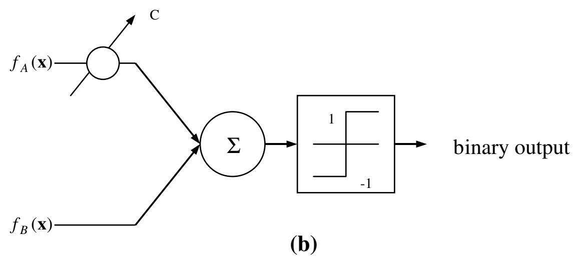


Fig. 3 The **a** pattern unit; and **b** output/decision unit of the PNN classifier

PNNs have been shown to learn up 2,00,000 times faster than BPNN (Patra et al. 2002). Since every training pattern needs to be stored, the features of being simple and fast come at the expense of larger memory requirements. This will not be a severe disadvantage if the PNN is implemented in a parallel hardware structure where memory is relatively inexpensive.

The general regression neural network (GRNN)

Originally discovered by Donald Specht (1991), the GRNN is a memory-based network that provides estimates of continuous variables and converges to the underlying regression surface. The GRNN, also belonging to a one-pass learning algorithm with a highly parallel structure, is capable of approximating any arbitrary function from historical data. The learning method learns near instantaneously since it simply stores patterns it has seen before and processes them through a nonlinear smoothing function to determine the component output PDF.

The foundation of GRNN operation is essentially based on the theory of nonlinear (kernel) regression. As a one-pass learning algorithm, main advantages of GRNN are that training only requires a single processing surface, and it is guaranteed to approach the Bayes' optimal decision boundaries as the number of training samples increases. The GRNN topology primarily consists of four layers: input, pattern, summation, and output, as shown in Fig. 4. The basic equation describing a GRNN output with m inputs ($\mathbf{x} \in \Re^m$) and one output ($y_i \in \Re^1$) is

$$y(\mathbf{x}) = \frac{\sum_i w_{ij}\theta_i(\mathbf{x})}{\sum_i \theta_i(\mathbf{x})}, \tag{21}$$

where $\theta_i\,(\mathbf{x})$ represents an arbitrary radial basis function. The Gaussian radial basis function is usually employed:

$$\theta_i(\mathbf{x}) = \exp\left[-\frac{(\mathbf{x}-\mathbf{x}_i)^{\mathrm{T}}(\mathbf{x}-\mathbf{x}_i)}{2\sigma^2}\right], \tag{22}$$

where the variables are defined as

$\mathbf{x}$:	input vector of predictor variables to GRNN;
$\mathbf{x}_i$:	training vector represented by pattern neuron i;
w_{ij}:	output related to $\mathbf{x}_i$;
σ:	the smoothing parameter.

If y_i are individual real-valued scalars, Eq. 21 is exactly Specht's GRNN which incorporates each and every training vector pair $\{\mathbf{x}_i \rightarrow y_i\}$ into its architecture, where $\mathbf{x}_i$ is a single training vector in the input space and y_i is the associated desired scalar output.

GDOP approximation and classification using neural networks

All input and output variables are normalized in the range [0, 1] to reduce the training time. The Hecht–Nielson's approach for the effectiveness of BP in learning complex, multidimensional functions was employed (Simon and El-Sherief 1995a), which was based on Kolmogorov's Theorem and extended to neural networks. 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 within the range [0, 1]:

$$\mathrm{fn} : [0,1]^m \subset \Re^m \rightarrow \Re^n. \tag{23}$$

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

$$\mathrm{GDOP} = (\lambda_1^{-1} + \lambda_2^{-1} + \lambda_3^{-1} + \lambda_4^{-1})^{1/2}. \tag{24}$$

The mapping is performed by defining the four variables

$$h_1(\vec{\lambda}) = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = \mathrm{trace}(\mathbf{H}^{\mathrm{T}}\mathbf{H}), \tag{25a}$$

$$h_2(\vec{\lambda}) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \lambda_4^2 = \mathrm{trace}[(\mathbf{H}^{\mathrm{T}}\mathbf{H})^2], \tag{25b}$$

$$h_3(\vec{\lambda}) = \lambda_1^3 + \lambda_2^3 + \lambda_3^3 + \lambda_4^3 = \mathrm{trace}[(\mathbf{H}^{\mathrm{T}}\mathbf{H})^3], \tag{25c}$$

$$h_4(\vec{\lambda}) = \lambda_1\lambda_2\lambda_3\lambda_4 = \det(\mathbf{H}^{\mathrm{T}}\mathbf{H}), \tag{25d}$$

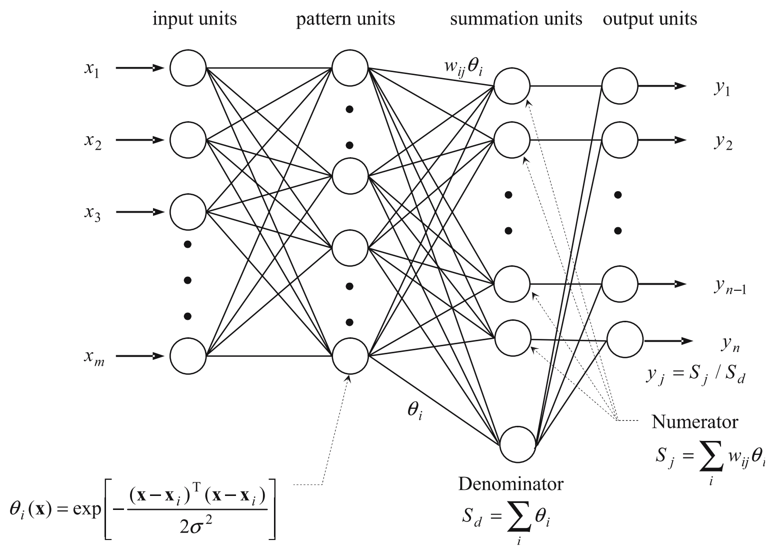
the $\vec{\lambda}^{-1}$ can be viewed as a functional $\Re^4 \rightarrow \Re^4$ mapping from $\vec{h}$ to $\vec{\lambda}^{-1}$ (Type 1 mapping), i.e., $\vec{\lambda}^{-1} = \mathrm{fn}(\vec{h})$:

Input: $(x_1, x_2, x_3, x_4)^{\mathrm{T}} = (h_1, h_2, h_3, h_4)^{\mathrm{T}}$,
Output: $(y_1, y_2, y_3, y_4)^{\mathrm{T}} = (\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1}, \lambda_4^{-1})^{\mathrm{T}}$.

Alternatively, the GDOP can be viewed as a functional $\Re^4 \rightarrow \Re^1$ mapping from $\vec{h}$ to GDOP (Type 2 mapping), i.e., $\mathrm{GDOP} = \mathrm{fn}(\vec{h})$:

Input: $(x_1, x_2, x_3, x_4)^{\mathrm{T}} = (h_1, h_2, h_3, h_4)^{\mathrm{T}}$,
Output: $y = \mathrm{GDOP}$.

Fig. 4 General GRNN architecture



The input–output relationships for the two types of mappings using NNs are shown in Fig. 5. The mapping (from $\vec{h}$ to $\vec{\lambda}^{-1}$ or from $\vec{h}$ to GDOP) is highly nonlinear and cannot be determined analytically but can be precisely approximated by the NN. The NN in this paper is designed to perform the Type 2 mapping.

The GDOP classifier can be employed for selecting one of the acceptable subsets. In the present study, the NN classifier has two types of output based on the GDOP factors. 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 $\vec{h}$ has four input variables h_i, which are normalized to the range [0,1]. The output has 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], representing the small, medium, and large GDOP, respectively. Similarly, classification into more groups is feasible based on the same philosophy.

Simulation and discussion

The BPNN and GRNN were employed for function approximation, while all the four types of networks (BPNN, OI net, PNN and GRNN) were employed for group classification based on their network characteristics, as summarized in Table 1.

Simulation was conducted using a Pentium III 733 MHz computer. The computer code was constructed by use of the Matlab® 6.1 version software. The receiver was simulated to be located at the top of the building of the Department, which has the approximate position of North 25.15°, East 121.78°, at an altitude of 62 m ($[-\ 3042329.2\quad 4911080.2\quad 2694074.3]^T$ m in WGS-84 ECEF coordinate) for 24 h duration. The GDOP was computed every 1 min, and collected in two data files every other minute, one file for training while the other for testing purpose. Consequently, there are 720 patterns for both the training and testing procedures. The solid dark line in Fig. 6a represents the GDOP solutions by matrix inversion.

GDOP approximation performance

GDOP function approximation performance using BPNN and GRNN is presented in this subsection. Figures 6 and 7 present the performance using BPNN. Good accuracy can be achieved by using BPNN for performing the GDOP function approximation (i.e., Eq. 24) given enough training time, e.g., 20,000 epochs of training for convergence in this study. After 20,000 epochs of training, increasing the number of hidden layer neurons only slightly improves the performance. In Fig. 6, the GDOP solutions by BPNN approximation using 10 and 50 hidden-layer neurons are compared to the one by matrix inversion (using Eq. 5). The GDOP residual is defined as the difference between the GDOP value by NN approach and by matrix inversion (denoted as 'MI'):$\mathrm{GDOP_{NN}} - \mathrm{GDOP_{MI}}$. The root-mean-squared error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{k=1}^{n} (y_{\mathrm{NN}} - y_{\mathrm{MI}})^2}{n}} \qquad (26)$$

is used as the error measure for evaluating approximation performance. The subscripts represent the results from NN approach and from matrix inversion, respectively. Figure 7 shows the RMSE versus the number of training iterations. Five numbers of hidden layer neurons (i.e., 10, 20, 30, 40, and 50) have been utilized. It is seen that the accuracy and the number of hidden layer neurons do not closely correlated until sufficiently large number of training iterations have been achieved. For example, 20,000 iterations of training are required for the present case.

Figures 8 and 9 demonstrate the GDOP approximation performance using GRNN. Six numbers of training patterns (i.e., 720, 360, 180, 90, 45, and 23) have been used. These numbers were selected based on the following two principles: (1) the numbers are descended by a 50% decreasing rate starting from 720; (2) all the training patterns in the smaller group must have also been included in the larger group. In Fig. 8, GRNN approximation performance is provided. Figure 8a gives

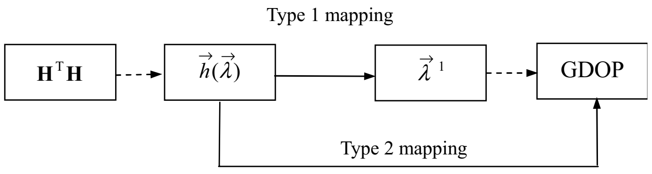


Fig. 5 The input-output relationships for two types of mappings using NNs

Table 1 Feasibility on classification and approximation for four types of NNs

	BPNN	OI Net	PNN	GRNN
Approximation	Yes	No	No	Yes
Classification	Yes	Yes	Yes	Yes

the results based on GRNN with 23 training patterns and that based on matrix inversion. The result from GRNN shows that very good accuracy can be achieved given that enough training patterns are used, which can be seen from Fig. 8b. Figure 9 provides a plot showing the RMSE versus the number of training patterns. Increasing the number of training patterns also increases the memory for software implementation and increases the structure complexity for hardware implementation and a trade-off in selecting the number of training patterns is required.

GDOP classification performance

Classification performance is presented in this subsection. When the GDOP threshold is set at 1.5 (This threshold value is chosen for ensuring a good satellite subset is obtained since GDOP $\leq$ 2 is a good value in GPS applications. Furthermore, the selection of the threshold does not closely relate to the training procedure and performance.), 127 samples among 720 are smaller than the threshold. The learning rate and momentum for the BPNN are 0.5 and 0.6, respectively. Five different training iterations are selected, which are 1,000, 5,000, 10,000, 15,000, and 20,000, respectively. The rates of correct classification for different number of training iterations are summarized in Table 2 and plotted in Fig. 10. As stated before, one important drawback for the BPNN classification is the long training time required. Improvement by increasing the number of hidden layer neurons is not significant after more than 20 neurons are used. The convergence speed becomes very slow after 10,000 training iterations. In the present work, even for the simplest case of 10 hidden layer neurons with 1,000 learning iterations, several minutes to hours (depends on the algorithms) is required for completing the training.

The other three NNs to be employed can significantly reduce the training time. For the OI Net, the parameters used include a fitting parameter $\rho = 0.1$, 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 the parameters. For the PNN, the correct classification rates have better accuracy when σ is small, which reaches the peak value approximately at $\sigma = 5e - 4$. Figure 11 presents the correct classification versus the number of training patterns using PNN. It is seen that if a sufficiently large number of training patterns is applied, very good classification accuracy can be achieved. Since the GRNN is basically an extended version of the PNN,

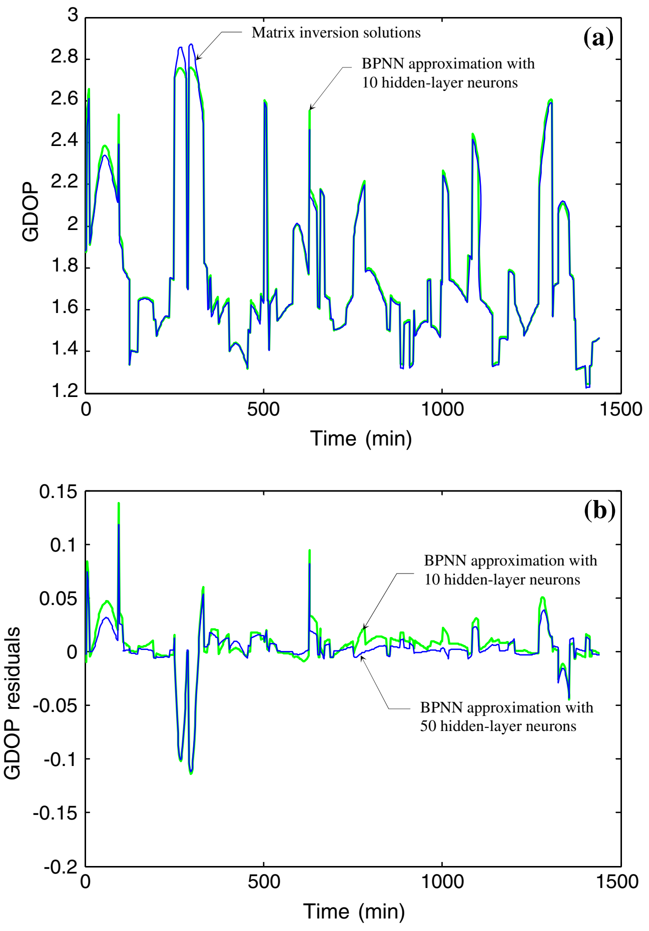


Fig. 6 **a** GDOP approximation performance using BPNN; and **b** comparison of residuals for 10 and 50 hidden-layer neurons after 20,000 epochs of training

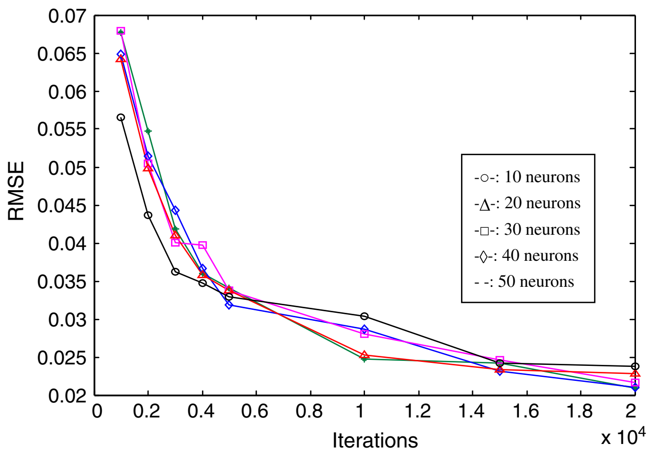
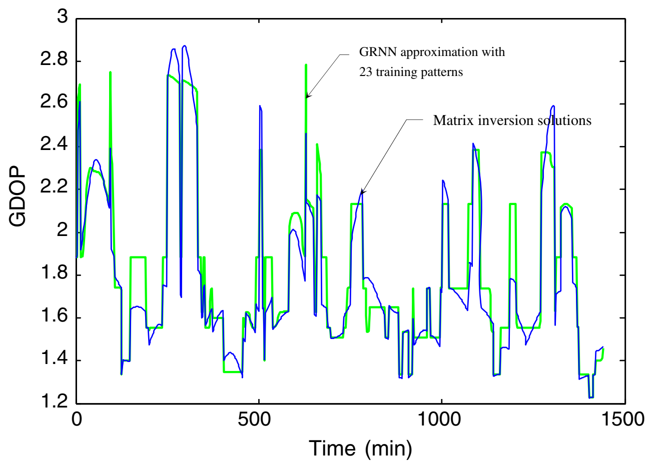


Fig. 7 RMSE of GDOP versus number of iteration using BPNN



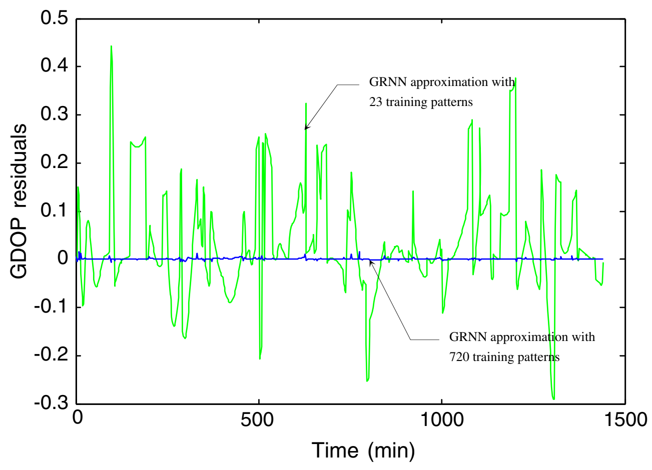


Fig. 8 **a** GDOP approximation performance using GRNN; and **b** comparison of residuals for 23 and 720 training patterns

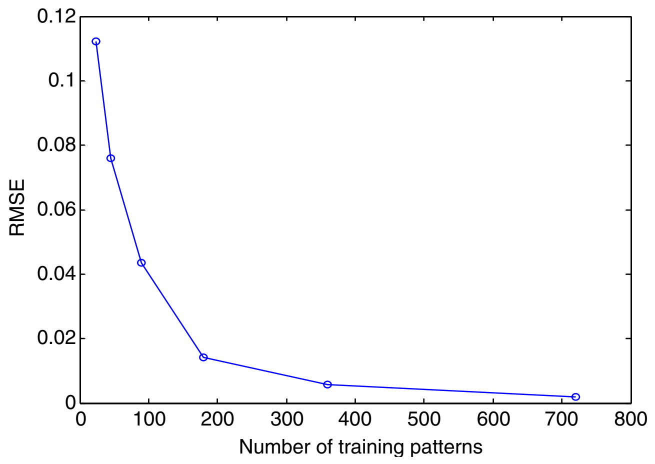


Fig. 9 RMSE of GDOP versus number of training patterns using GRNN

the classification performance by GRNN is essentially same as that by PNN. However, the minimum number of training patterns required for the PNN to work is normally less than that for the GRNN. The results on classification rate and training time for the four NNs are summarized in Table 3. The accurate classification rates of OI Net are in the range of 93–99%, with the training times around 0–3 s (depending the number of training inputs). The accurate classification rates of PNN and GRNN are in the range 93–100%, with the training times no more than 1 s.

Conclusions

The NN-based GDOP approximation and classification have been successfully conducted. The performances have been explored and discussed. Two types of NN approximators (BPNN and GRNN) and four types of NN classifiers (BPNN, OI Net, PNN, and GRNN), respectively, were focused on. While it is recognized that the BPNN has been most popular throughout all neural applications, it is also well known to have some drawbacks, especially on slow learning. The other three NNs employed in this paper can overcome this problem. From the viewpoint of accuracy, all the networks are able to provide sufficiently good performance, given enough time (for BPNN) or enough training data (for the other three networks). Consequently, selection of the NNs involves a tradeoff between user's requirements.

Table 2 Classification performance using the BPNN

No. of neurons in hidden layer	Correct classification rates (%) for varying training iterations				
	1,000 iterations	5,000 iterations	100,000 iterations	150,000 iterations	200,000 iterations
10	93.89	97.08	99.31	99.31	99.58
20	95.00	99.31	99.17	99.44	99.58
30	95.28	99.31	98.89	99.31	99.31
40	95.14	99.31	99.44	99.44	99.31
50	95.14	99.31	99.17	99.31	99.31

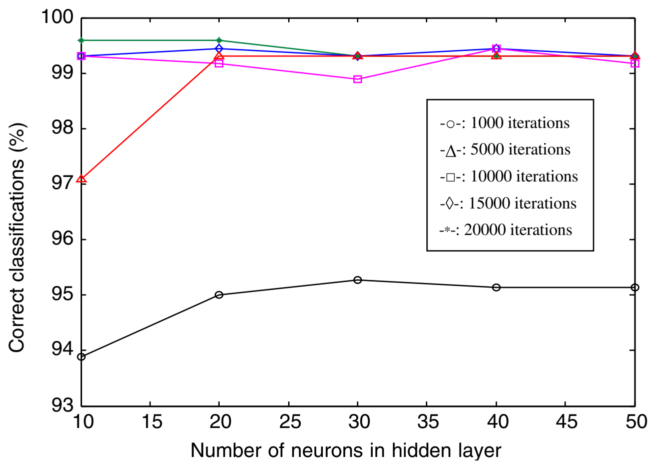


Fig. 10 Rates of correct classifications versus number of hidden layer neurons by BPNN

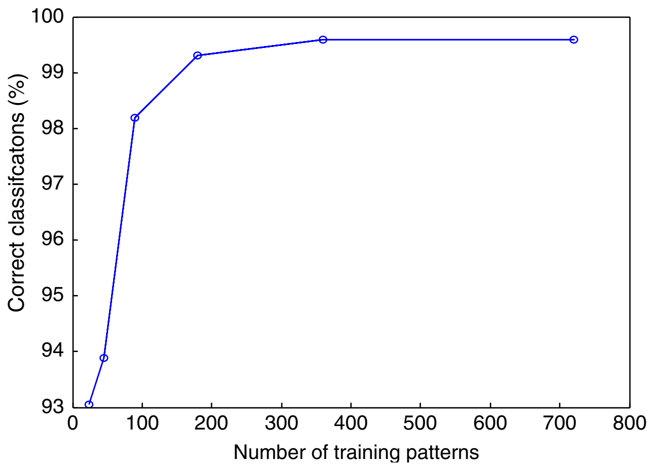


Fig. 11 Rates of correct classifications versus number of training patterns using PNN

Table 3 Comparison of classification rate and training time based on four types of NNs

	BPNN	OI Net	PNN	GRNN
Correct classification rate	93–100%	93–99%	93–100%	93–100%
Training time	Minutes–hours	0–3 s	0–1 s	0–1 s

Acknowledgments This work has been supported in part by the National Science Council of the Republic of China through grant no. NSC 94-2212-E-019-003. Valuable suggestions and detailed comments by the reviewers are gratefully acknowledged.

References

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

Defigueiredo RJP (1983) A generalized Fock space framework for nonlinear system and signal analysis. IEEE Trans Circ Syst CAS-30:637–647

Defigueiredo RJP (1990) A new nonlinear functional analytic framework for modeling artificial neural networks. In: Proceedings of 1990 international symposium on circuits and systems, New Orleans, pp 723–726

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

Jwo DJ, Chin KP (2002) Applying back-propagation neural networks to GDOP approximation. J Navig 54(2):97–108

Kihara M, Okada T (1984) A satellite selection method and accuracy for the global positioning system. Navig 31(1):8–20

Patra PK, Nayak M, Nayak SK, Gobbak NK (2002) Probabilistic neural network for pattern classification. In: Proceedings of 2002 international joint conference on neural networks, IJCNN '02, pp 1200–1205

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

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

Sin SK, defigueiredo RJP (1992) An evolution-oriented learning algorithm for the optimal interpolative net. IEEE Trans Neural Netw 3(2):315–323

Specht DF (1988) Probabilistic neural networks for classification, mapping, or associative memory. In: IEEE international conference on neural networks, pp 525–532

Specht DF (1991) A general regression neural network. IEEE Trans Neural Netw 2(6):568–576

Stein BA (1985) Satellite selection criteria during Altimeter aiding of GPS. Navig J Inst Navig 32(2):149–157

Widrow B, Lehr MA (1990) 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. Proc IEEE 78(9):1415–1442

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