Nested Hyper-Rectangle Learning Model for Remote Sensing: Land Cover Classification

Li Chen

Abstract
This study presents an exemplar-based nested hyper-rectangle learning model (NHLM) which is an efficient and accurate supervised classification model. The proposed model is based on the concept of seeding training data in the Euclidean m-space (where m denotes the number of features) as hyper-rectangles. To express the exceptions, these hyper-rectangles may be nested inside one another to an arbitrary depth. The fast and one-shot learning procedures can adjust weights dynamically when new examples are added. Furthermore, the “second chance” heuristic is introduced in NHLM to avoid creating more memory objects than necessary. NHLM is applied to solving the land cover classification problem in Taiwan using remote sensed imagery. The study investigated five land cover classes and clouds. These six classes were chosen from field investigation of the study area according to previous study. Therefore, this paper aims to produce a land cover classification based on SPOT HRV spectral data. Compared with a standard back-propagation neural network (BPNN), the experimental results indicate that NHLM provides a powerful tool for categorizing remote sensing data.

Introduction
This study presents a machine-learning model, nested hyper-rectangle learning model (NHLM) which is based on exemplar learning. This supervised learning method is used to classify land cover from remote sensor data. The model is derived from a learning model that was proposed originally as a human learning model. In the original theory, training data are stored in memory verbatim as points, and their representation does not change. Restated, a data point in Euclidean m-space, where m is the number of variables or features in an example (e.g., in E_m space) is defined within the model as a vector of feature values plus a label that representing the example category. Furthermore, its special data structures of axis parallel rectangle (two points on the diagonal of each rectangle) require limited computer memory. Unlike generalization processes using symbolic formulae, such as the regression approach by more general classes, the NHLM algorithm modifies hyper-rectangles by growing and reshaping them in a well-defined fashion (see Expanding Hyper-rectangles). In this study, NHLM first processes the experimental results of a series of remotely sensed data. Notably, a trained NHLM can reproduce the data for which it was trained. The generalization capacity of a trained NHLM can approximate the experimental results of similar data.

There are several types of classification procedures, i.e., supervised, unsupervised and hybrid classifications. The fundamental difference between supervised and unsupervised methods is that the former involves a training step, which is followed by a classification step (Schowengerdt, 1997). It includes, e.g., the minimum-distance-to-mean classifier, parallelepiped classifier, and the Gaussian maximum likelihood classifier (Tso and Mather, 1999). Bayesian statistical theory has been used extensively as a theoretically robust foundation to classify remotely sensed data. To achieve maximum a posterior (MAP) estimation both class-conditional and prior probabilities must be modeled. However, it is difficult to model the prior information, and the prior probability density function (PDF) is typically assumed as being uniform. MAP estimation then reduces to maximum likelihood (ML) estimation, which relies on estimates of the mean vector and variance-covariance matrix for each class. These estimates are generally derived from spectral attributes, and are applied on a pixel-by-pixel basis. In many cases, the ML classifier yields acceptable results. However, failure to use prior information limits classification accuracy (Tso and Mather, 1999). The ML classification is based on the assumption that the probability distribution for each class is a multivariate normal distribution. The basic drawback of the ML classification is that a distribution is assumed for the input data, and there is no learning involved in the ML (Kishore and Patnaik, 2000). The parallelepiped classifier method (Lillesand and Kiefer, 1994) is a traditional supervised classification of the remote sensing. Although it is the fastest and most efficient classifier in the computing sense, it is not the most widely used because it has several disadvantages. The most important disadvantage is that it can leave many unclassified pixels. Another disadvantage of this classification method is that it can have large overlap between training pixels. On the other hand, NHLM could solve the non-classification problem by choosing the nearest exemplar. Moreover, because of the error tolerance NHLM could avoid generating the large hyper-rectangle instead of many small ones.

The artificial neural network (ANN) is a recently popular nonparametric approach to supervised classification. ANN has been extensively applied to perform classification of remotely sensed data (Atkinson and Tatnall, 1997). ANN has been shown to be able to map land cover more accurately than the widely used statistical classification techniques. The multilayer perceptron (MLP) with back-propagation (BP) algorithm might be one of the most widely used models for classification. Compared to the conventional statistical classifier, the BPNN is distribution-free and non-parametric,
and is more robust. However, the biggest drawback to this method is the large training times necessary for mean square error minimization (Paola and Schowengerdt, 1995). Besides, BPN exhibits some serious drawbacks such as slow convergence in learning phase, the potential convergence to local minimum, the common chaotic behavior, and the inability to detect over-fitting.

Adaptive resonance theory (ART) has supported the evolution of a series of real-time neural network models for unsupervised and supervised learning and pattern recognition. The first ART model, ART-1 (Carpenter and Grossberg, 1987) was an unsupervised learning system for categorizing binary input patterns. ART-2 (Carpenter and Grossberg, 1991) and fuzzy ART (Lang and Witbrock, 1988; Carpenter et al., 1997) extend the binary ART-1 domain to be able to categorize both analog and binary input patterns. Recently, ART has been used without complementary coding of remote sensing data overcoming the limits of coding (Gamba and Houshmand, 2001). The ideas that underlie all ART systems is a pattern matching process that compares the current input with a selected representation of a learned category, or active hypothesis. NHLM and ART networks (Carpenter et al., 1997) are compared as follows.

- **ART** is based on an iterative neural network with adjusted weights. On the other hand, NHLM is a case-based learning model. They fundamentally differ.
- The coding schemes of ART differs greatly from that of NHLM are very different. Each ART input is complement coded, with an interval between 0 and 1. This complement coding doubles the number of dimensions of input variables and must be normalized to the values of the input variables.
- ART has more parameters than NHLM to be determined.
- ART uses the choice function based on iterative fuzzy inferences including reset and match tracking. Restated, NHLM can be more concisely and efficiently implemented and can be comprehended more concretely.
- The output of ART must also be normalized, and corresponds to a category probability distribution. It must use a set of complex computations to obtain an output at the testing stage.

This paper develops an alternative method to ML or ANN, called NHLM. It is investigated to determine its efficiency and accuracy in solving remote sensing problems. The next section details the characteristics of NHLM followed by the training algorithm. A case study in which a remote sensing problem is solved using NHLM, which is a case study of the classification of land coverage, using remote sensing data. Finally, a summary is presented and conclusions.

### Characteristics of NHLM

The exemplar-based learning (EBL) algorithm represents one likely future of machine learning. The strategy of the exemplar-based learning (EBL) algorithm is based on storing examples represented as points in Euclidean m-space, \( E^m \) and then comparing the new example to these stored examples to find the most similar example (Medin and Schaffer, 1978). The algorithm is a very powerful tool for categorizing tasks. The NHLM enables the points to be generalized into axis-parallel hyper-rectangles (Thornton, 1987). As the generalizations grow large, some exceptions may arise, and create “holes” in the hyper-rectangles. These may, in turn, have holes inside them, resulting in a nested structure of hyper-rectangles. The creation of these holes allows NHLM to include additional exceptions inside the hyper-rectangles, nested as deeply as required by the data. Figure 1 shows H2 and H3, which are nested structures in the two-dimensional feature space.

Helmholdt et al. (1989) proved some very strong optimal results for their algorithm. Salzberg (1989) applied their model to three different domains: to predict the recurrence of breast cancer, to classify iris flowers, and to predicting the survival times of heart attack patients. Salzberg demonstrated that NHLM performs as well as, or better than, other algorithms, which were applied to the same data sets in all cases.

Many important characteristics of this model are summarized as follows:

**Representation**

Several systems establish rules through learning processes; however, such rules cannot exhaust the possible representation of the knowledge that learning may yield. However, NHLM generates a memory space filled with exemplars from the system’s experience. Other shapes that could be examined in future research include spheres, ellipses, and convex hulls. However, the implementation considered here involves nested hyper-rectangles created by NHLM.

**Problem Domain**

NHLM is most suitable for domains in which the exemplars form clusters (ideally, convex solids) in feature space, and whose behavior of the exemplars in a cluster is similar constant. However, if the exemplars are strung out along an infinite curve, then the best description of the domain is the equation of that curve (which can be obtained using a multiple regression method), rather than a set of exemplar objects. Although NHLM can be applied in such a domain, it will not generate nearly as concise a description as would a curve-fitting method.

**Partitioning Feature Space**

Using a Euclidean distance formula to determine distance, basically partitions the feature space among the hyper-rectangles. The partitioning of the plane is substantially more complex than that induced by a pair of points or circles. NHLM also attaches a weight to each rectangle which further complicates the shape of the surface.

**Discrete and Continuous Variables**

NHLM can handle variables that are binary, discrete, or continuous, so it is independent of domain. The model uses an “error tolerance” parameter for continuous variables, which indicates how close two values must be in order to
be considered as “matching”; therefore, the continuous variables are approximated by a discrete set of values. A high error tolerance supports broad generalization, the coarseness categories, and abstract representations.

The Nested Hyper-rectangle Learning Algorithm

The basic algorithm of NHLM is that it uses some existing events as a foundation to predict the outcomes of other events by building the structure of hyper-rectangles. As learning is completed, all inputs coded by a category access it directly, a search is automatically disengaged, and the performance rate reaches 100 percent on the classification training set. This process includes adjusting the weights of the model’s parameters in time, and making the system learn. The main procedures are described as below.

Seeding

For the sake of making predictions, NHLM must have a history of examples as the bases of its predictions. These examples are chosen at random from the training set. Memory is initialized by “seeding” it with a small set of examples (the minimum size of this set is one). The seeding process simply stores each example in memory without attempting to make any prediction.

Matching

After initialization, every new example is matched to memory by a matching process. This process uses the distance metric to measure the distance (or similarity, on the contrast) between a new data point (an example) and an exemplar (an existing point or hyper-rectangle in E\textsuperscript{d}). The best match is the one, which has the smallest distance metric value which is then used to make a prediction in the way that the new example has the same category as the closest matching exemplar.

Let the new example be \(E\) and the existing hyper-rectangle be \(H\). The match score between \(E\) and all exemplars \(H\) is stored in memory. It is calculated by measuring the Euclidean distance between the two objects. The distance metric is determined as follows:

\[
D_{EH} = W_0 \sum_{i=1}^{m} \left( \frac{W_i (E_i - H_i)}{\text{Max}_i - \text{Min}_i} \right)^2
\]

(1)

where \(W_i\) is the weight of the exemplar \(H\), \(W_i\) is the weight of the feature \(i\), \(E_i\) is the value of the \(i\)th feature in example \(E\), \(H_i\) is the value of the \(i\)th feature in exemplar \(H\), \(\text{Min}_i\), \(\text{Max}_i\) are the minimum and maximum values of that feature, and \(m\) is the number of features.

Let \(H_{\ell}\) be the lower end of the range, and \(H_{\bar{u}}\) be the upper end, then the distance metric becomes:

\[
E_i - H_i \begin{cases} 
E_i - H_{\ell} & \text{if } E_i > H_{\ell} \\
H_{\bar{u}} - E_i & \text{if } E_i < H_{\ell} \\
0 & \text{otherwise}
\end{cases}
\]

(2)

The distance measured by this formula is equivalent to the length of a line dropped perpendicularly from the point \(E\) to the nearest surface, edge, or corner of \(H\). Note that points internal to a hyper-rectangle have distance 0 to that rectangle. Figure 2 illustrates these different distance measurements. Furthermore, a point is assigned to the innermost rectangle if it is internal to several nested rectangles, because the inner rectangles serve as exceptions to surrounding rectangles. In Figure 1, point \(E\) belongs to rectangle \(H_1\), because of \(H_1\) is the inner rectangle. In the simplest case, the system predicts that the dependent variable for a new example will have the same value at that stored on the closest exemplar. Exemplars have properties such as weights, shapes, and sizes, all of which can be adjusted based on the results of the prediction. Learning only occurs when NHLM gets feedback about its classification.

The formula divides distances along each dimension by \((\text{Max}_i - \text{Min}_i)\) in order to standardize them to the interval [0,1] so that each feature in the exemplars has same basis. Any other standardizing method that can be replaced this procedure. For example, \(\frac{E_i - H_i}{\text{Max}_i - \text{Min}_i}\) can be replaced by the function \(\Phi(E_i) - \Phi(H_i)\) where \(\Phi\) represents the cumulated probability of normal distribution.

There are two weights in the distance metric, \(W_0\) and \(W_i\). \(W_0\) is a simple measure of how frequently the exemplar, \(H\), has been used to make a correct prediction. \(W_i\) is the ratio of the number of times \(H\) has been used to the number of times it has resulted in a correct prediction. Consequently, it can indicate the reliability of each exemplar. Thus if a hyper-rectangle \(H\) is used many times, but nearly always makes the wrong prediction, the weight \(W_0\) will grow very large, and \(H\) will tend not to be chosen as the closest match in the future. If \(H\) is a noisy point, then it will eventually be ignored as \(W_0\) increases. The minimum value of \(W_0\) is unity in the case of perfect prediction, shown in Equation 3.

\[
W_0 = \frac{N_t}{N_e}
\]

(3)

where \(N_t\) = the total number of times an exemplar has been used, and \(N_e\) = the number of time an exemplar makes the correct prediction.

Error Tolerance

The error tolerance of a dependent variable indicates how close two values must be in order to be considered equivalent. This error tolerance is necessary because for real-valued variables, it is usually the case that two values never match exactly, and yet the system needs to know if its prediction.
was close enough to be considered correct (Salzberg 1991). For discrete variables, the error tolerance could set to be zero. We expanded the application of error tolerance to determine the neighborhood of the input of each hyper-rectangle. It is useful in the discrete case. In addition, an error tolerance of input variables is also set automatically to restrict the expansion size of hyper-rectangles in this paper.

**Expanding Hyper-rectangles**
When a new input data \( E \) matches an existing hyper-rectangle \( H \) and under the error tolerance of the dependent variable, then \( H \) will expend to include \( E \) with the smallest hyper-rectangle \( H \oplus E \).

**Learning**
The other weight measure, \( W_i \), is the weight of the \( i^{th} \) feature. These weights are adjusted to reflect the fact that all features do not normally have equal importance in a category decision. NHLM adjusts the weights \( W_i \) on the feature \( i \) after discovering that it has made the wrong prediction. Weight \( W_i \) adjustment is executed in a very simple loop: for each \( f_i \) if \( E_f \) matches \( H_f \), the weight \( W_i \) is decreased by setting \( W_i = W_i(1 - \Delta f) \), where \( \Delta f \) is the global feature adjustment rate. An increase in weights causes the two objects to seem farther apart; if \( E_f \) does not match \( H_f \), then \( W_i \) is increased by setting \( W_i = W_i(1 + \Delta f) \). The \( \Delta f \) normally uses as 0.05. With higher values of \( \Delta f \), accurate rates are lower.

**Second Chance**
If the system makes the wrong prediction, it has one more chance to make the right one. This “second chance” heuristic is used by NHLM in order to avoid creating more memory objects than necessary. The idea is to try very hard to make a generalization and thus keep down the size of memory. So, before creating a new exemplar, NHLM first looks at the second best match in memory. Assume here that \( H_1 \) was the closest exemplar to \( E \) (\( D_{D1} \) was smallest) and \( H_2 \) was second closest exemplar; i.e., \( H_2 \) would be the closest if \( H_1 \) were removed. If \( H_2 \) will give the correct prediction, then the system tries to adjust hyper-rectangle shapes to make the second closest exemplar into the closest exemplar. It does this by creating a generalization from \( H_2 \) and \( E \). The goal of this process is to improve the predictive accuracy of the system without increasing the number of exemplars stored in memory. The flow chart of NHLM combining with second chance is shown in Figure 3.

**NHLM Training Procedures**
During training, input vectors \((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\); \( n \) is the number of input data and \( x \) includes \( m \) independent variables.

1. **Seeding**
   - Weights: \( w_i = 1.0, i = 1 \ldots m \)
   - \( H_{1x} = x^{(1)} \) and \( H_{1y} = y^{(1)} \)  
     (Creating a point rectangle)
   - \( Y_{1y} = y^{(1)} \)
   - \( e = 1 \)
   (current index of rectangle)

2. **Matching**
   - For \( i = 2 \) to \( n \) do
     - \( e1 = \) nearest of all \( H \) to \( x^i \)
       (using Equations 1 and 2)

3. **Second Chance**
   - \( e2 = \) second of all \( H \) to \( x^i \)
     (using Equations 1 and 2)
   - If [neighbor \((x^i, e1))/[neighbor means under the error tolerance]]

4. **Learning**
   - Adjusting \( Wh_{e1} \) by Equation 3, correct times + 1
   - End else
   - If [neighbor \((x^i, e2)]
     - Begin
       - Expanding \( e2 \) to \( e2 \oplus x^i \)
       - Adjusting \( Wh_{e2} \) by Equation 3, incorrect
       - For \( j = 1 \ldots m \) do
         - begin
           - If distance\((e1j, xj) < \) distance \((e2j, xj)\), \( Wj = Wj(1 - \Delta f) \)
           - Else
             - If distance\((e1j, xj) > \) distance \((e2j, xj)\), \( Wj = Wj(1 + \Delta f) \)
         - end
       - End else
     - Begin
       - \( c = c + 1 \)
       - \( H_{c2} = x^{(c)} \) and \( H_{cy} = y^{(c)} \)
         (Creating a point rectangle)
       - \( Y_{yc} = y^{(c)} \)
       - \( Wh_c = 1.0 \)
     - end

Figure 3. The learning procedure of NHLM.
Applications to Remote Sensing Data: Classifying Land Coverage

The case study concerns a problem of land cover classification in Taiwan. The purpose of the image classification procedure is automatically to categorize all pixels in an image into land cover classes.

Study Area and Data Set

The study area is centered on the Tseng-Wen Reservoir watershed in southern Taiwan, and covers an area of 481 km² (Plate 1). The main types of land cover in the watershed include indigenous forest, scrub, villages, orchards (mango, litchi, papaya, and plum), water, and betel nut plantations. Table 1 lists the five land cover classes obtained from a report and the clouds. The goal of this study is to present a system for classifying land cover using SPOT HRV spectral data. Accordingly, SPOT HRV data concerning the watershed area of Tseng-Wen Reservoir were collected by the Department of Agriculture Engineering, National Taiwan University on 08 November 1996 (Cheng et al., 1998). The multi-spectral SPOT consists of green (R2, 0.50 to 0.59 μm), red (R3, 0.61 to 0.68 μm), and near-infrared (R4, 0.78 to 0.89 μm) bands. The SPOT images were corrected for atmospheric conditions (Cheng et al., 1998).

Image Classification

The NHLM was applied to classify the land cover in of one remote sensing image. The same data were processed using a supervised back-propagation neural network (BPNN). Plate 1 shows the 5,027 pixels selected from the total study area, and the ground reference data were obtained by field investigation. The sampling of data chosen process was based on a method used by Cheng et al. (1998) which involved divergence-based feature selection. MSS imagery obtained from the SPOT satellite and the textural features of the images were used to classify land-use in the Tseng-Wen Reservoir watershed.

Cross Validation

All data were grouped in two sets called the training set and the testing set. When the training process had been

Table 1. Definition of Information Classes with Number of Pixels Extracted from the Ground Image

<table>
<thead>
<tr>
<th>Class</th>
<th>Information Class</th>
<th>Number of Pixels (Total = 5027)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Water Body</td>
<td>1371</td>
</tr>
<tr>
<td>2</td>
<td>Betel Nut</td>
<td>1275</td>
</tr>
<tr>
<td>3</td>
<td>Building</td>
<td>197</td>
</tr>
<tr>
<td>4</td>
<td>Cloud</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>Orchard</td>
<td>544</td>
</tr>
<tr>
<td>6</td>
<td>Woods</td>
<td>1440</td>
</tr>
</tbody>
</table>

Plate 1. (a) The false colored image of the study area (Cheng et al. 1998). (b) The distribution of sampling data of the six classes.
completed, the constructed model was used to predict the output values for the data in the testing set (which the process had never seen).

Therefore, the use of these remote sensing data in the learning by NHLM depends on splitting the 5,027 pixel data must be split into at least two groups: (1) The first group is used for training the model which is called the training set, and (2) the second group is used to measure the performance of the model which is called the testing set.

After the training stage is completed, the NHLM model’s performance is then estimated. The set model is applied to N randomly selected new data without modifying or changing the model structure or parameters to predict Y values (representing one of the six classes). These predicted Y values are compared to the real values. Several values of n were used to determine the effect of the number of training data on the accuracy of classification. In each case, 100 runs using randomly selected sets of training data were conducted to match their Y values by using the NHLM with various numbers of training data in the model, and the overall accuracy was recorded. The procedures were performed 100 times independently to support statistical analysis.

The data were thus randomly classified as follows: (1) Training data: several training data set’s values of \( n = 10, 50, 100, 500, 1000, 2000, 3000 \) and 4000, and (2) Testing data: \( N = 1027 \) (had never used these output values in the training stage).

**Results and Discussion**

The same data were selected for use in the training and testing stages to compare the performance of NHLM with that of BPN. The ability a BPN to discriminate among classes is specified by its weights of connections. During training, they are iteratively adjusted to establish a configuration that enables the network to distinguish the prototype patterns of interest. The weights of connections are modified according to the information it has learned to train the network. The network learns by comparing its output with each input pattern with a target output for that pattern, and then calculating the error and propagating an error function backward through the net. The values of the input parameters in the project are presented to the network to enable it to be run after it has been trained. The network then calculates the node output using the existing weights and the thresholds developed during training. The BPN with the delta-bar-delta algorithm (Jacobs 1988) have some combinations of neural parameters that have been set by trials. The gradient descent algorithms include the following steps (Richards 1993):

1. Select training pixels for each class and specify the desired output vector.
2. Initialize weights as random numbers between 0 and 1 (Typically small values near zero are used.).
3. Set the frequency of the updating of weights after all training pixels in all classes have been seen.
4. Propagate the training data forward through the net, one pixel at a time.
5. Calculate the output and accumulate the total error relative to the desired output.
6. Change the weights after all training pixels have been used.
7. Repeat Steps 4 to 6 until an acceptable tolerance level is reached.

BPN were trained using various parameters, initial conditions, and numbers of hidden units. The process was terminated after 1,000 iterations of the BPN. Figure 4 compares the average of overall accuracies obtained using 100 runs of NHLM and BPN. The results indicate that as the number of training data increases to 4,000, the average overall accuracy of NHLM increases to 0.733. However, when the number of training data exceeds 1000, the average overall accuracy of BPN reaches its limit bounds of 0.635. Briefly, NHLM is more efficient and accurate than BPN in classifying land cover. Additional comparisons between NHLM and BPN are described below.

**Confusion Matrix**

The average testing result of these 100 runs through NHLM and BPN (both \( n = 4000, N = 1027 \)) were demonstrated as confusion matrix shown in Table 2 and 3. The overall accuracy in Table 2 equals 73.3 percent. Whereas Table 3, the overall accuracy is 63.5 percent. Table 2 indicates producer’s accuracies range from just 31.2 percent (“Building”) to 97.2 percent (“Water Body”) and user’s accuracies range vary from 40.7 percent (“Orchard”) to 96.8 percent (“Water Body”). It is obvious that the producer’s accuracy of water body is the highest among all categories; in contrast, the identification of building is very difficult. Betel nut and woods are easy to confuse with each other. The poor classification result of “building” is due to it confusing with betel nuts or orchards. These buildings are fragmentary and located in the mountain area with surrounding variable vegetation, so the misclassification is caused by the resolution of input data. Moreover, the best overall accuracy of the 100 runs’ classification results produced by NHLM and BPN are 75.6 percent and 64.8 percent, respectively.

**Kappa Statistic**

To account for errors of commission and errors of omission, Cohen’s Kappa coefficient of agreement was used (Cohen, 1960; Lillesand and Kiefer, 1994). With values from Table 2, a Kappa coefficient of was produced 0.645 (for NHLM) and from Table 3, \( \hat{k} = 0.518 \) (for BPN). Confidence intervals around Kappa can be computed using sample variance and the fact that the Kappa statistic is asymptotically normally distributed. Equation 4 is the approximate large sample
statistical significance of any given matrix or the differences among matrices:

\[ H_0 : \kappa_1 = \kappa_2 \quad \text{versus} \quad H_1 : \kappa_1 \neq \kappa_2, \]

\[ Z = \frac{\kappa_1 - \kappa_2}{\sqrt{\text{Var}(\kappa_1) + \text{Var}(\kappa_2)}}. \quad (5) \]

The \( Z \) score between NHLM and BPN equals 4.15, which is larger than \( Z_{0.05} = 1.96 \). Therefore, the difference of results between NHLM and BPN is significant.

**Conclusions**

This study presents a nested hyper-rectangle learning model (NHLM), which is a flexible and efficient method for managing the classification problems. The primary advantage of this learning model is that it partitions the input space into localized regions, thus facilitating the construction of a local system model. The learning model was robust in the face of noise and incomplete data. One of the strengths of NHLM is the simplicity of both the algorithm and the representation it creates. Moreover, another important feature of NHLM is the fact that humans can easily interpret the hyper-rectangles as a decision making tool.

The proposed model was applied land cover classification based on SPOT HRV spectral data. According to the learning curve of average overall accuracies through 100 runs by cross validation, NHLM performance improves with increasing training data. The confusion matrix and Kappa statistic results demonstrate that NHLM is a powerful estimation tool for remote sensing image categorization. The performance of land cover classification using NHLM is more

### Table 2. The Confusion Matrix of Average 100 Runs by NHLM

<table>
<thead>
<tr>
<th>Ground References</th>
<th>1 Water Body</th>
<th>2 Betel Nut</th>
<th>3 Building</th>
<th>4 Cloud</th>
<th>5 Orchard</th>
<th>6 Woods</th>
<th>Row Total</th>
<th>User’s Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>NHLM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Water Body</td>
<td>271.1</td>
<td>0</td>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>8.3</td>
<td>280.2</td>
<td>96.8</td>
</tr>
<tr>
<td>2 Betel Nut</td>
<td>1.7</td>
<td>157.2</td>
<td>14.5</td>
<td>1.7</td>
<td>24.2</td>
<td>67.5</td>
<td>266.8</td>
<td>58.0</td>
</tr>
<tr>
<td>3 Building</td>
<td>0</td>
<td>2.9</td>
<td>12.8</td>
<td>0</td>
<td>6.4</td>
<td>0</td>
<td>22.1</td>
<td>57.9</td>
</tr>
<tr>
<td>4 Cloud</td>
<td>0</td>
<td>0</td>
<td>2.2</td>
<td>36.4</td>
<td>5.5</td>
<td>0</td>
<td>44.1</td>
<td>82.5</td>
</tr>
<tr>
<td>5 Orchard</td>
<td>0</td>
<td>23.2</td>
<td>8.4</td>
<td>4.9</td>
<td>38.3</td>
<td>19.3</td>
<td>94.1</td>
<td>40.7</td>
</tr>
<tr>
<td>6 Woods</td>
<td>6.2</td>
<td>58.7</td>
<td>2.3</td>
<td>0</td>
<td>15.6</td>
<td>236.9</td>
<td>319.7</td>
<td>74.1</td>
</tr>
</tbody>
</table>

**Column Total** 279 242 41 43 90 332 1027 51.8

**Producer’s Accuracy %** 97.2 65.0 31.2 84.7 42.6 62.7 71.4 0.000454646** 73.3***

*Kappa value = 64.5 percent  
**Variance of Kappa = 0.000454646  
***Overall Accuracy = 73.3 percent

### Table 3. The Confusion Matrix of Average 100 Runs by BPN

<table>
<thead>
<tr>
<th>Ground References</th>
<th>1 Water Body</th>
<th>2 Betel Nut</th>
<th>3 Building</th>
<th>4 Cloud</th>
<th>5 Orchard</th>
<th>6 Woods</th>
<th>Row Total</th>
<th>User’s Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 Water Body</td>
<td>238</td>
<td>0.9</td>
<td>1.6</td>
<td>1</td>
<td>0</td>
<td>13.1</td>
<td>254.6</td>
<td>93.5</td>
</tr>
<tr>
<td>2 Betel Nut</td>
<td>16.9</td>
<td>144.4</td>
<td>13.4</td>
<td>3.4</td>
<td>29.8</td>
<td>81.8</td>
<td>289.7</td>
<td>49.8</td>
</tr>
<tr>
<td>3 Building</td>
<td>2.3</td>
<td>5.5</td>
<td>8.1</td>
<td>0</td>
<td>7.3</td>
<td>0</td>
<td>23.2</td>
<td>34.9</td>
</tr>
<tr>
<td>4 Cloud</td>
<td>3.2</td>
<td>5.5</td>
<td>8.1</td>
<td>0</td>
<td>7.3</td>
<td>0</td>
<td>23.2</td>
<td>34.9</td>
</tr>
<tr>
<td>5 Orchard</td>
<td>5.5</td>
<td>17.7</td>
<td>13.7</td>
<td>0</td>
<td>21.4</td>
<td>208.3</td>
<td>311.2</td>
<td>66.9</td>
</tr>
<tr>
<td>6 Woods</td>
<td>15.1</td>
<td>62.7</td>
<td>3.7</td>
<td>0</td>
<td>21.4</td>
<td>208.3</td>
<td>311.2</td>
<td>66.9</td>
</tr>
</tbody>
</table>

**Column Total** 279 242 41 43 90 332 1027 64.5

**Producer’s Accuracy %** 85.3 59.7 19.8 70.5 26.0 62.7 0.000481769** 63.5***

*Kappa value = 51.8 percent  
**Variance of Kappa = 0.000481769  
***Overall Accuracy = 63.5 percent

The variance for Kappa, and Equation 5 is used to calculate significance (i.e., \( Z \) score):

\[
\text{Var}(\kappa) = \frac{1}{N} \left[ \theta_1 (1-\theta_1) + \frac{2(1-\theta_1)(2\theta_1 - \theta_1 - \theta_2)}{(1-\theta_2)^3} + \frac{(1-\theta_1)^2 - 4\theta_1^2}{(1-\theta_1)^2} \right]
\]

where

\[
\begin{align*}
\theta_1 &= \sum_{j=1}^{r} \frac{X_{ij}}{N} \\
\theta_2 &= \sum_{j=1}^{r} \frac{X_{ij}X_{-j}}{N^2} \\
\theta_3 &= \sum_{j=1}^{r} \frac{X_{ij}(X_{ij} + X_{-j})}{N^2} \\
\theta_4 &= \sum_{j=1}^{r} \frac{X_{ij}^2(X_{ij} + X_{-j})}{N^3}
\end{align*}
\]

where \( X_{ij} \) = diagonal subtotal for row \( i \), column \( j \), \( X_{ij} \) = row subtotal for row \( i \), \( X_{ij} \) = column subtotal for column \( i \), \( X_{ij} \) = observed cell value at row \( i \), column \( j \), and \( X_{ij} \) = column subtotal for column \( j \).
efficient and accurate than when using backpropagation network (BPN). Moreover, the learning time of NHLM is significantly less than that of BPN, because NHLM is based on a one-shot learning process. In contrast, BPN requires considerable iteration. NHLM requires more memory than BPN, although the difference is minimal.

Acknowledgments
The author would like to thank Prof. K.S. Cheng and National Taiwan University for the data collection.

References


(Received 30 June 2003; accepted 03 November 2003; revised 19 December 2003)