Feature Evolution for Classification of Remotely Sensed Data

Demetris Stathakis and Kostas Perakis

***Abstract*—In a number of remote-sensing applications, it is critical to decrease the dimensionality of the input in order to reduce the complexity and, hence, the processing time and possibly improve classification accuracy. In this letter, the application of genetic algorithms as a means of feature selection is explored. A genetic algorithm is used to select a near-optimal subset of input dimensions using a feed-forward multilayer perceptron trained by backpropagation as the classifier. Feature and topology evolution are performed simultaneously based on actual classification results (wrapper approach).**

***Index Terms*—Feed-forward neural networks, genetic algorithms, image classification, remote sensing.**

I. Introduction

The PURPOSE of this letter is to establish a methodology for the application of genetic algorithms in classifying remotely sensed data using neural networks by performing, concurrently, feature selection and topology identification. Genetic algorithms are deployed since they are known to have both the theoretical foundations and the practical capacity of revealing near-optimal solutions [4], [6]. It appears that when it comes to determining neural-network parameters, including topology, authors of even recent papers are still struggling with the same trial-and-error methodology that has been used for the past 17 years despite the progress noted in fields other than remote sensing. The novelty of this letter mainly consists in bridging this chasm. Furthermore, the coding that we use is novel, enabling the concurrent evolution of topology with feature selection which has been a major handicap in the past.

We begin by briefly reviewing the literature on feature selection by standard (sequential) and genetic-algorithm-based methods in Section II. In Section III, we present the experiment and results of feature evolution performed in the context of this letter. Brief conclusions on the proposed method are given in Section IV.

II. Feature Evolution

Feature selection can be formulated as a search problem [19]. A straightforward way of coding this problem is to use a binary string with length equal to the number of available input features, with the value of one indicating the presence of a particular feature and zero showing its absence [19]. The easiest way is to keep the architecture of the neural network fixed, but this is not efficient since a different number of inputs (features) would normally require a different topology for optimal classification accuracy. With or without fixing the topology, a major obstacle faced when using neural networks as the classifier is the noisy fitness-evaluation problem [18], i.e., the fact that the evaluation of a specific set of input features is also dependent on the random initialization parameters. In other words, the same input dimensions produce sometimes quite different outputs because of the different random weights assigned to the links of the neural network.

The unconstrained combinatorial optimization version of feature selection corresponds to finding the subset of input dimensions that result into the lowest error rate. The constrained version is more interesting, because it calls for simultaneously minimizing the error as well as the number of input dimensions [14]. It is also more difficult because it belongs to the category of multiobjective problems [10, Ch. 2], since the minimization of both the number of features and, at the same time, of the classification error is sought [4, Ch. 5]. Thus, the notion of optimality is not obvious, and it might not be wise to combine both criteria into a single number because the two targets are qualitatively different.

In a pioneer experiment [14], the authors classified a number of samples taken from aerial photography using the k-nearest neighbors method. They used a canonical genetic algorithm to solve the constrained version of feature selection. They introduced a formula that we will refer to as the Siedlecki–Sklansky throughout this letter. This function favors the reduction of features rather than the reduction of error. This seems to be the only valid objective in the original context that the formula was developed, i.e., for large-scale feature selection (>20 input dimensions), where the number of features has to be drastically reduced. It consists of the actual number of features plus a penalty value to reward classification accuracy

$$f_{(a)} = l_{(a)} + \frac{e^{\frac{(1-r_{(a)})-t}{m}} - 1}{e^1 - 1}, \qquad \text{for} \quad l_{(a)} = 0 \text{ we set } f_{(a)} = 0 \tag{1}$$

where $a = (a_1, \ldots, a_n)$ is a solution string and $l_{(a)}$ is the number of features (the number of 1s in the string). Overall accuracy (r) is normalized in [0, 1]. The original publication uses the error rather than the accuracy rate, but the remote-sensing community is used to the accuracy figures, hence, this slight modification. The feasibility threshold t is the level of error that is considered feasible. The tolerance margin m is a scale factor that controls the steepness of the function.

Manuscript received August 14, 2006; revised January 8, 2007.

D. Stathakis is with the Joint Research Centre, the European Commission, Institute for the Protection and Security of the Citizen, AGRIFISH, MARS-FOOD, TP266, 21020 Ispra, Italy (e-mail: demetris.stathakis@jrc.it).

K. Perakis is with the Department of Urban and Regional Planning Engineers, University of Thessaly, Pedion Aeros, 38334 Volos, Hellenic Republic (e-mail: perakis@uth.gr).

Digital Object Identifier 10.1109/LGRS.2007.895285

Symbol e denotes the base of the natural logarithm. The only possibility for this function to become negative is for zero number of features selected. In this case, we choose to simply set the fitness function to zero. The original publication adopts a slightly different approach [14, eq. (3.4)]. Note that

$$f_{(a)} \in [0.418, +\infty] \qquad \forall l_{(a)} > 0 \tag{2}$$

given that the penalty value

$$\frac{e^{\frac{\left(1-r_{(a)}\right)-t}{m}} - 1}{e-1} \in [-0.582, +\infty]. \tag{3}$$

They find that the genetic algorithms perform better than the sequential-selection methods. They also conclude that the time required by the genetic algorithm to discover the solution is comparable to that of the sequential-search method. Finally, they suggest that the genetic algorithms perform better in the large-scale feature selection case because they exhibit approximately linear increase of time complexity.

Subsequent results with the genetic algorithms and the Siedlecki–Sklansky function are controversial. The authors in [7] evaluated this function. Synthetic aperture radar data and 18 features (textures) are used. It is difficult to make a conclusive statement on their results because the solutions presented have different number of features. The genetic algorithm solution is 1/10 less accurate than the sequential-floating-forward-selection solution, but at the same time, the latter has 1/3 less features. To say which of the two solutions is preferable is clearly problem-dependent. It is also evident, however, in their discussion that they had difficulties in determining the (t, m) parameters of the fitness function.

In [3], the authors find that the genetic algorithms usually perform slightly worse than the sequential-floating-forward-selection method. The performance, however, is heavily dependent on the two parameters (t, m) of the Siedlecki–Sklansky function. They also find contradictory results with respect to the scalability of genetic algorithms compared to the Siedlecki–Sklansky [14] paper. They observed that the performance of genetic algorithms decreases with the increase of input dimensions. In both of these studies [3], [7], it cannot be concluded with confidence that the poor performance is due to genetic algorithms themselves or due to the specific implementation of the fitness function used.

The authors in [20] use a genetic algorithm with binary string representation for feature selection with hyper spectral satellite imagery and compare the two classification methods: the standard and fuzzy versions of k-nearest neighbors. They deploy the Siedlecki–Sklansky fitness function to solve the constrained feature-selection problem. They find that the same levels of classification accuracy can be achieved by significantly reducing the number of input features used. It is evident in their results that the genetic algorithms outperform the sequential-floating-forward-selection method.

Very few are the cases where the genetic algorithms have been used in feature selection for land-cover classifications with satellite images. Stathakis and Kanellopoulos [16] used a canonical genetic algorithm with binary representation and direct encoding to automatically determine the optimum structure of the neural network [16]. Recently, Kavzoglu and Mather [9] compared feature selection using the genetic algorithms to several conventional methods. A binary representation of input dimensions is adopted. Paradoxically, the number of sought features is fixed because they experimentally find that a certain number (eight) of input dimensions is likely to provide optimal accuracy. The architecture of their multilayer perceptron (MLP), trained by the backpropagation [12], [17], is also kept fixed (8 : 10 : 7). They adopt a filter technique [9], i.e., the fitness function is measured using several class separability indexes based entirely on the training data without any reference to the classification results. The obvious benefit of this approach is reduction in the required processing time. A major drawback, however, is that the genetic algorithm may be successful in optimizing the separability of classes based on the index used but may not be as successful based on the actual classification results. There might be some evidence of this side effect in their work (e.g., compare results in [9, Tables 3 and 4]).

III. Method and Results

As the basis for comparison, the sequential-floating-forward-selection method is used. A canonical genetic algorithm to solve the constrained version of feature selection is then deployed. The slightly modified Siedlecki–Sklansky fitness function, described in the previous section, is adopted. The point of departure in this letter is that the topology of the neural network is concurrently evolved with feature selection. Other important novelties of our approach include the adoption of constrained optimization, without fixing, however, the sought number of features, as well as the implementation of the wrapper approach by evaluating the actual classification results of the neural networks.

Our data set refers to Lefkas Island in the western part of the Hellenic Republic. Inputs to the system are seven Landsat-7 Enhanced Thematic Mapper Plus (ETM+) bands; plus elevation, slope, and aspect, which are derived from Shuttle Radar Topography Mission data [13]. The Landsat scene was acquired in July 2000. The complete input features are named [B, G, R, NIR, MIR1, TIR, MIR2, DEM, SLP, and ASP], accordingly. Outputs are five COoRdinate INformation on the Environment (CORINE) Level 1 land-use classes [2], viz. artificial surfaces, agricultural areas, forest and seminatural areas, wetlands, and water bodies. Following the recommendation of Congalton [1], we acquire 50 samples per class using stratified sampling. The CORINE land-cover data set is used as the reference to annotate the output vector. We split the total number of reference points (250) into two equal-in-size parts. One of them serves as the training data set, whereas the other one as the testing data set. A more detailed description of the data used can be found in Stathakis and Vasilakos [15].

A. Sequential Floating Forward Selection

Sequential floating forward selection is the one of the sequential variants that competes and sometime outperforms the genetic optimization of features [3], [7]. Fisher's linear discriminant ratio is used, here, as the measure of performance.

By maximizing the Fisher criterion, the distance between the means of the classes is maximized while the variance within each class is minimized. In the two-class case, the Fisher criterion becomes

$$J(w) = \frac{|m_a - m_b|^2}{s_a^2 + s_b^2} \tag{4}$$

where m represents the mean, s^2 represents the variance, and the subscripts denote the two classes a and b.

This method produces rapidly [TIR, ASP, NIR, B, G, SLP, MIR1, R, MIR2, and DEM] as the ordered output vector. The associated cost per solution is [4.0, 6.2, 8.4, 10.7, 12.7, 14.4, 15.7, 17.3, 19.0, and 20.2], respectively. Features are placed in this vector, from left to right, according to the performance. Thus, the best five inputs are [TIR, ASP, NIR, B, and G]. The best overall classification out of ten runs, using this five-input combination and exactly the same neural-network settings that are used with the genetic algorithm, is 74.4% and 70.4% for 32 and 36 hidden nodes, respectively.

B. Feature Selection via a Genetic Algorithm

1) Genetic and Neural Parameters: We adopt the binary representation to code the input dimensions, which is quite standard in the literature and similar to the method used in Kavzoglu and Mather [9]. The value of one declares the presence of a particular input dimension in the solution, whereas the value of zero declares its absence. The difference to Kavzoglu and Mather [9] is that we do not fix the sought number of classes. We let, instead, the genetic algorithm decide which the optimal number of inputs is and at the same time decide which the optimal combination of inputs is. An additional advantage to this strategy is that a correction mechanism, forcing the number of input dimensions to be fixed that might introduce unpredictable behavior, is not used. Moreover, the choice to have a variable network topology rather than a fixed one is made. Rosenblatt suggests [11, Ch. 25] that a neural-network topology, which is evolved, with connections created or disregarded based on demonstrated utility, rather than fixed, might lead to better and more compact topologies. He also suggests that, by observing the terminal states reached by the evolved system, we might gain an insight of the dynamics of the system, which can be used for improving future designs. The topology is evolved here using an indirect encoding, i.e., specifying only the number of nodes in the single hidden layer.

In summary, the binary string (chromosome) representation has the form shown on Fig. 1, where the first ten bits correspond to an equal number of input dimensions. The following five bits correspond to the number of nodes in the hidden layer. The search is limited in five bits, i.e., range of integers [0–31]. Based on the Kanellopoulos–Wilkinson rule [8], which states that the optimal number of hidden nodes in a neural network with a single hidden layer having ten inputs and five outputs should be between 10 and 40 nodes, we add the constant number of eight to this five-bit figure so that the range of hidden nodes becomes [8, 40].

Other genetic-algorithm parameters include population size of 20, two-point crossover at a rate of 0.9, and Gaussian muta-

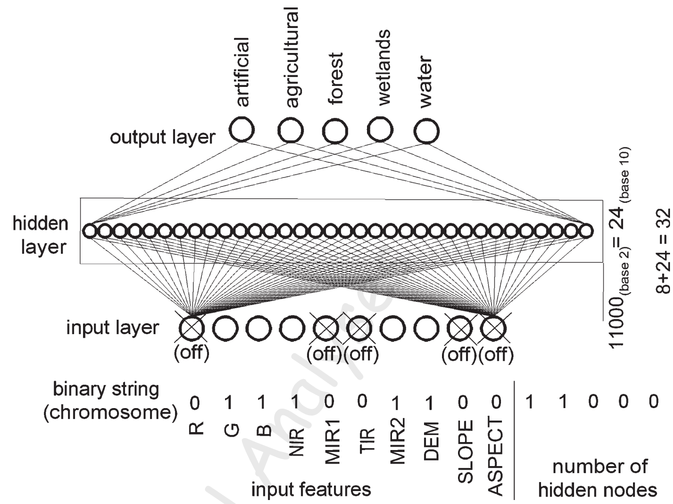

Fig. 1. Genetic-algorithm representation and mapping on to a corresponding neural network. The network is fully connected, but note all the connections are shown here. Zero values in the string denote inactive input nodes (OFF). The number of hidden nodes is the sum of the constant eight plus the five-bit binary number converted to decimal.

tion. The two best individuals are copied unaltered in the next generation (elitism). The termination criterion is the completion of 100 generations. Each individual's performance is evaluated for selection based on its rank rather than its actual score. The population is split into five groups, and the best individuals per group are selected for reproduction (tournament selection).

The classification is performed using a standard fully connected feed-forward MLP. Learning rate is set to 0.03 and momentum is 0.9. Log sigmoid transfer function is used. The performance function for training the neural network is the mean of squared errors. Termination criterion is either the completion of 110 training epochs or reaching an error level of 0.02 mean-square error for the training data set. The number of epochs is the one that is more often met. The winner-takes-all rule is applied to the output of the neural network to obtain a single output class per input vector. For faster learning, we use the Levenberg–Marquardt backpropagation variant [5].

2) Constrained Feature Evolution: The objective is to optimize accuracy while at the same time minimize the number of input features. The Siedlecki–Sklansky fitness function is used with feasibility threshold $t = 0.145$ and tolerance margin $m = 0.02$. Previous experience with the data suggests that a feasible overall testing-classification range for this data using different methods should be 85%–95%. For this reason, we choose $(t,\ m)$ combinations adapted for this accuracy range. We preserve the t value originally proposed [14] value, but we slightly modify the m value to cover for the expected range of accuracies. If the accuracy level (m) is set far above what can be achieved in practice, then the information gained by the fitness function is not sufficient to drive the evolution. If m is a bit higher (0.02) than originally proposed, the transition is smoother, which might be preferable.

The best performing string, 0111001100 | 11000, contains five features [G, B, NIR, MIR2, and DEM] and 32 nodes in the hidden layer with fitness 4.53 and overall accuracy of 88.8%. This solution is discovered in the third generation and lost after that. The accuracy matrix for this solution is presented in Table I

TABLE I
ACCURACY MATRIX FOR BEST TOPOLOGY AND FEATURE COMBINATIONS: (TOP) TRAINING AND (BELOW) TESTING. SIMPLE DIRECT CODING WITH POPULATION = 20. PRODUCER'S AND USER'S REFER TO PERCENT ACCURACY

training

Khat = 0.97	ATF	WET	FOR	WAT	AGR	User's
ATF	25	0	0	0	0	100
WET	0	24	0	0	1	96
FOR	0	0	22	0	1	95.7
WAT	0	0	0	22	0	100
AGR	1	0	0	0	29	96.7
producer's	96.2	100	100	100	93.5	97.6

testing

Khat = 0.88	ATF	WET	FOR	WAT	AGR	User's
ATF	19	0	1	0	0	95
WET	2	24	0	0	0	92.3
FOR	0	0	25	0	2	92.6
WAT	0	0	0	28	0	100
AGR	3	2	2	0	17	70.8
producer's	79.2	92.3	89.3	100	89.5	90.4

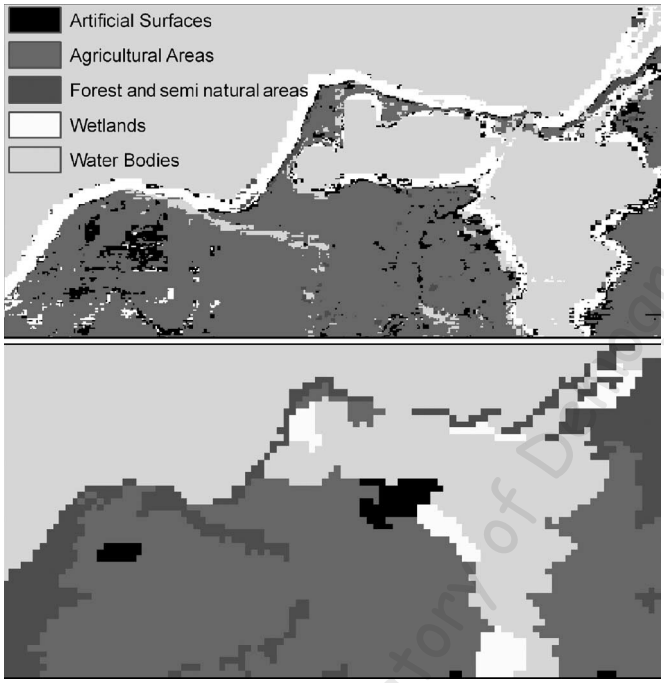

Fig. 2. (Top) Partial view of the actual classified image compared to (bottom) the target land use CORINE data set.

and a part of the actual classified image in Fig. 2. The second best string, with fitness of 4.58 and overall accuracy of 88%, contains the same five input features, 36 hidden nodes, and is discovered after nearly 3/4 of total number of generations. The third best string contains the same five features, yields fitness of 4.67 and overall accuracy of 87.2, it is discovered in 16 different generations, most of which are at the late stages of evolution. The same accuracy is produced by both a 32- and 36-node solution.

By deploying a genetic algorithm that simultaneously searches for the optimal feature combination as well as for the optimal number of nodes in the hidden layer, we obtain results that are more accurate than using standard neural networks [15] than using several neuro-fuzzy configuration [15], and slightly more accurate than those obtained using neural networks with evolved topology but without feature selection [16] for the same data set. Feature evolution is also compared favorably to standard methods such as sequential floating forward selection. Method comparison is presented in Table II.

TABLE II
METHOD COMPARISON FOR THE SAME DATA SET

Method	overall testing classification accuracy
Evolved topology and features with NN topology 5:32:5 (proposed method)	90.4%
SFFS (Fisher) with NN topology 5:32:5	74.4%
Standard NN with manual feature selection 5:15:5 [15]	80.8%
Evolved topology without feature selection [16]	88.0%
Using all input features with NN topology 10:32:5 (best results over 10 runs with 10:32:5 and 10:36:5)	83.2%

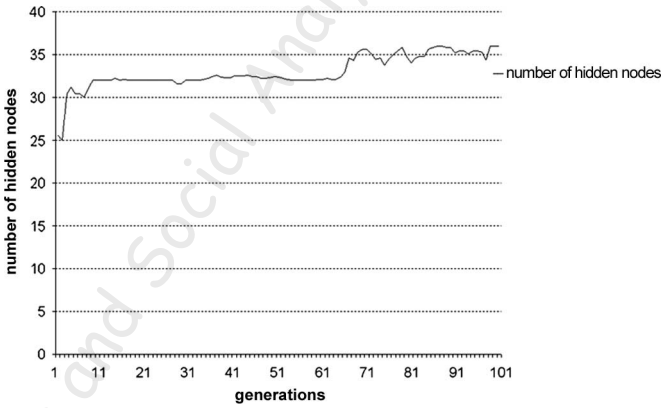

Fig. 3. Average number of nodes in the hidden layer per generation during optimization with the genetic algorithm.

By observing the terminal states of feature evolution, we realize that the optimal number of hidden nodes based on the knowledge discovered by the genetic algorithm is around 35, as shown on Fig. 3. The discovery of this number of hidden nodes is interesting, because it proves to be well outside the Kanellopoulos–Wilkinson-rule-derived topology. The optimal number of features stabilized after approximately half of available training generations to five, as shown on Fig. 4.

The time requirements of the sequential search compared to those of the genetic algorithm are negligible. It takes approximately 20 h to complete 100 generations on a dual 2.80-GHz processor with 1.5-GB of RAM computer configuration. Sequential floating forward selection produces the results nearly instantly.

IV. CONCLUSION

In this letter, a very efficient method of designing neural networks via genetic optimization is described. The proposed method solved the two main problems faced in any neural-network-classification problem: which input features and which hidden topology to use. Note also that the proposed method can incorporate in the optimization string additional parameters that have to be set, provided that (parallel) computational power

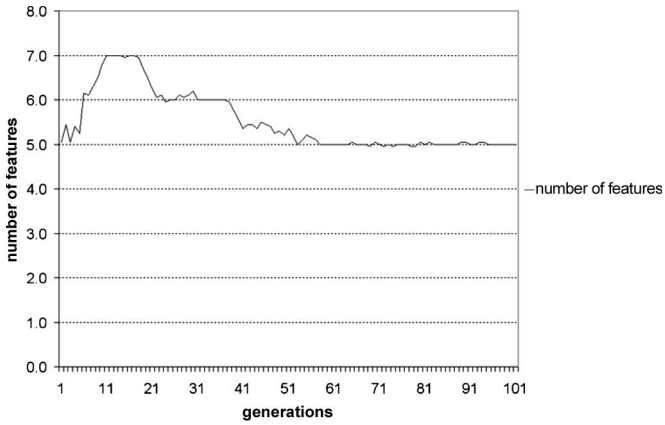

Fig. 4. Average number of features per generation during optimization with the genetic algorithm.

is available. We elaborate upon applying the commonly used Siedlecki–Sklansky fitness function. An insight on how to select the proprietary parameters of this function is also provided. The results show that the concurrent evolution of features and network topology can yield more accurate results. It takes significant amount of time to run the process, but it might be still preferable compared to the trial-and-error approach. Future work could include experiments with large-scale feature selection as well as additional datasets.

Finally, the Siedlecki–Sklansky fitness function imposes two parameters (t, m) that need to be set and that have nothing to do with the original genetic-algorithm parameters. In addition, for applications such as classification for some land-cover mapping, it might be more appropriate to target accuracy maximization rather than number of features minimization. It follows that the design of a new fitness function is desirable that favors a solution with fewer features only when the accuracy level is the equal or higher.

ACKNOWLEDGMENT

The intellectual property rights of the satellite image used belong to IMAGE2000 of JRC, based on Landsat-7 ETM+ © ESA distributed by Eurimage; ortho-correction EU15 © Metria. CORINE is "© EEA, Copenhagen, 2000" downloaded from http://dataservice.eea.eu.int/dataservice/metadetails.asp?id=759.

REFERENCES

[1] R. Congalton, *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*. Boca Raton, FL: Lewis Publishers, 1998.
[2] COoRdinate INformation on the Environment (CORINE), *Program of the European Commission to Provide Consistent Information on Land Cover Across Europe*, 2000. [Online]. Available: http://image2000.jrc.it/
[3] F. Ferri, P. Pudil, M. Hatef, and J. Kittler, "Comparative study of techniques for large-scale feature selection," in *Pattern Recognition in Practice IV, Multiple Paradigms, Comparative Studies and Hybrid Systems*, E. S. Gelsema and L. S. Kanal, Eds. Amsterdam, The Netherlands: Elsevier, 1994, pp. 403–413.
[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[5] M. T. Hagan and M. Menhaj, "Training feed-forward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.
[6] J. Holland, *Adaptation in Natural and Artificial Systems*, 2nd ed. Cambridge, MA: MIT Press, 1992.
[7] A. Jain and D. Zongker, "Feature selection: Evaluation, application and small sample performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 153–158, Feb. 1997.
[8] I. Kanellopoulos and G. Wilkinson, "Strategies and best practice for neural network image classification," *Int. J. Remote Sens.*, vol. 18, no. 4, pp. 711–725, Mar. 1997.
[9] T. Kavzoglu and P. M. Mather, "The role of feature selection in artificial neural network applications," *Int. J. Remote Sens.*, vol. 23, no. 15, pp. 2919–2937, Aug. 2002.
[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1996.
[11] F. Rosenblatt, *Principles of Neurodynamics*. Washington, DC: Spartan, 1962, p. 616.
[12] D. E. Rumelhart and J. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA: MIT Press, 1986. PDP research group.
[13] *Shuttle Radar Topography Mission (SRTM)*, 2000. [Online]. Available: http://www2.jpl.nasa.gov/srtm/
[14] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognit. Lett.*, vol. 10, no. 5, pp. 335–347, Nov. 1989.
[15] D. Stathakis and A. Vasilakos, "A comparison of several computational intelligence based classification techniques for remotely sensed optical image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 44, no. 8, pp. 2305–2318, Aug. 2006.
[16] D. Stathakis and I. Kanellopoulos, "Evolving neural networks for classifying remotely sensed images," *Photogramm. Eng. Remote Sens.* submitted for publication.
[17] P. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA, 1974. unpublished.
[18] X. Yao and Y. Liu, "Towards designing artificial neural networks by evolution," *Appl. Math. Comput.*, vol. 91, no. 1, pp. 83–90, 1998.
[19] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.
[20] S. Yu, S. DeBacker, and P. Scheunders, "Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite images," *Pattern Recognit. Lett.*, vol. 23, no. 1–3, pp. 183–190, Jan. 2002.