ELSEVIER

Image and Vision Computing 21 (2003) 591–608

image AND vision COMPUTING

www.elsevier.com/locate/imavis

Genetic algorithm based feature selection for target detection in SAR images

Bir Bhanu*, Yingqiang Lin

Center for Research in Intelligent Systems, College of Engineering, University of California, B232 Bourns Hall, Riverside, CA 92521-0425, USA

Abstract

A genetic algorithm (GA) approach is presented to select a set of features to discriminate the targets from the natural clutter false alarms in SAR images. Four stages of an automatic target detection system are developed: the rough target detection, feature extraction from the potential target regions, GA based feature selection and the final Bayesian classification. A new fitness function based on minimum description length principle (MDLP) is proposed to drive GA and it is compared with three other fitness functions. Experimental results show that the new fitness function outperforms the other three fitness functions and the GA driven by it selected a good subset of features to discriminate the targets from clutters effectively.

Keywords: ATR system; Feature selection; Genetic algorithm; Minimum description length; Target detection

1. Introduction

Automatic detection of potential targets in SAR imagery is an important problem [1,2]. A constant false alarm rate (CFAR) detector is commonly used to 'prescreen' the image to localize the possible targets [2]. Generally, targets correspond to bright spots caused by strong radar return from natural or man-made objects. Parts of the imagery that are not selected are rejected from further computation. In the next stage of processing, regions of interest are further examined to distinguish man-made objects from natural clutter. Finally, a classifier such as a Bayesian classifier, a template matcher or a model-based recognizer is used to reject man-made clutter.

In general, the goal of feature selection is to find the subset of features that produces the best target detection and recognition performance and requires the least computational effort. Feature selection is important to target detection and recognition systems mainly for three reasons:

First, using more features can increase system complexity, yet it may not always lead to higher detection/recognition accuracy. Sometimes, many features are available to a detection/recognition system. However, these features are not independent and may be correlated. A bad feature may greatly degrade the performance of the system. Thus, selecting a subset of good features is important.

Second, features are selected by a learning algorithm during the training phase. The selected features are used as a model to describe the training data. Selecting many features means a complicated model being used to approximate the training data. According to the minimum description length principle (MDLP), a simple model is better than a complex model [21]. Since the training data may be corrupted with a variety of noises, a complex model may overfit the training data. Thus, a complex model may be sensitive to noises in the training data and its performance on unseen test data may be bad. In this paper, we use genetic algorithm (GA) to select as few features as possible to describe the training data effectively.

Third, using fewer features can reduce the computational cost, which is important for real-time applications. Also it may lead to better classification accuracy due to the finite sample size effect.

GAs are widely used in image processing, pattern recognition and computer vision [1,3,4]. They are used to evolve morphological probes that sample the multi-resolution images [5], to generate image filters for target detection [6], to select good parameters of partial shape matching for occluded object recognition [7], to perform

* Corresponding author. Tel.: +1-909-787-3954; fax: +1-909-787-3188.
E-mail addresses: bhanu@cris.ucr.edu (B. Bhanu); yqlin@cris.ucr.edu (Y. Lin).

0262-8856/03/$ - see front matter © 2003 Elsevier Science B.V. All rights reserved.
doi:10.1016/S0262-8856(03)00057-X

pattern clustering and classification [8], etc. GAs are also used to automatically determine the relative importance of many different features and to select a good subset of features available to the system [9].

The focus of this paper is to select a minimal set of features to distinguish targets from natural clutter. The approach is based on a closed loop system involving GA based feature selection and a Bayesian classifier. GA uses a MDLP-based fitness function that combines the number of features to be used and the error rate of the classifier. The results are presented using real SAR images. The experimental results show that the MDLP-based fitness function is the most effective in selecting a minimal set of features to describe the data accurately compared to other three fitness functions, and the subset of features selected by GA can greatly reduce the computational cost while at the same time maintaining the desired detection accuracy.

Section 2 presents the related research and the contribution of this paper. Section 3 describes the approach, feature evaluation criteria, fitness functions, the prescreener used to detect potential target regions, the features for target discrimination and the application of GAs to feature selection. Experimental results are presented in Section 4 and Section 5 provides the conclusions of the paper.

2. Related research

Bhanu and Lee [10] present a closed loop image segmentation system which incorporates a GA to adapt the segmentation process to changes in image characteristics caused by variable environmental conditions such as time of day, time of year, clouds, etc. The segmentation problem is formulated as an optimization problem and the GA efficiently searches the hyperspace of segmentation parameter combinations to determine the parameter set which maximizes the segmentation quality criteria in terms of edge-border coincidence, boundary consistency, pixel classification, object overlap and object contrast. Their experimental results demonstrate that GA can continuously adapt the segmentation process to normal environmental variations to provide robust performance when interacting with a dynamic environment. Emmanouilidis et al. [11] discuss the use of multi-criteria GAs for feature selection. With multi-criteria fitness functions, GA tries to minimize the number of features selected while maintaining the high classification accuracy. The algorithm is shown to yield a diverse population of alternative feature subsets with various accuracy and complexity trade-off. It is applied to select features for performing classification with fuzzy models and is evaluated on real-world data sets such as cancer data set in which each data point has nine input features and one output label (malignant or benign). Estevez and Caballero [12] propose a GA for selecting features for neural network classifiers. Their algorithm is based on a niching method to find and maintain multiple optima. They also introduce a new mutation operator to speed up the convergence of the GA. Rhee and Lee [13] present an unsupervised feature selection method using a fuzzy-genetic approach. The method minimizes a feature evaluation index which incorporates a weighted distance between a pair of patterns used to rank the importance of the individual features. A pattern is represented by a set of features and the task of GA is to determine the weighting coefficients of features in the calculation of weighted distance. Matsui et al. [14] use GA to select the optimal combination of features to improve the performance of tissue classification neural networks and apply their method to problems of brain MRI segmentation to classify gray matter/white matter regions.

Quilan and Rivest [15] explore the use of MDLP for the construction of decision trees. The MDLP defines the best decision tree to be the one that yields the minimum combined length of the decision tree itself plus the description of the misclassified data items. Their experimental results show that the MDLP provides a unified framework for both growing and pruning the decision tree, and these trees seem to compare favorably with those created by other techniques such as C4 algorithm. Gao et al. [16] use MDLP to determine the best model granularity such as the sampling interval between the adjacent sampled points along the curve of Chinese characters or the number of nodes in the hidden layer of a three layer feed-forward neural network. Their experiments show that in these two quite different settings the theoretical value determined using MDLP coincides with the best value found experimentally. The key point of their work is that using MDLP the optimal granularity of the model parameters can be computed automatically rather than being tuned manually.

In this paper, we use GA to select a good subset of features used for target detection in SAR images. The target detection task involves the selection of a subset of features to discriminate SAR images containing targets from those containing clutter. Our method is a novel combination of GA based optimization of a criterion function that involves classification error and the number of features that are used for the discrimination of targets from natural clutter in SAR images. The criterion (fitness) function we propose in this paper is based on the MDLP and it compares favorably with other three fitness functions. We assume the joint distribution of features follows Gaussian distribution. The criterion function is optimized in closed-loop with a Bayesian classifier evaluating the performance of each set of features. The GA used in feature selection is adaptive in the sense that it can automatically adapt the parameters such as crossover rate and mutation rate based on the efficiency of GA search in the feature space. As compared to this work, the feature selection presented in Refs. [2,17] for target vs. natural clutter discrimination measures exhaustively the performance of each combination of the features by the P_d (probability of detection) vs. P_{fa}

(probability of false alarm) plot produced by it. The higher the P_d and the lower the P_{fa}, the better the combination of features.

3. Technical approach

The purpose of the GA based feature selection approach presented in this paper is to select a set of features to discriminate the targets from the natural clutter false alarms in SAR images. The approach includes four stages: rough target detection, feature extraction from the potential target regions, feature selection based on the training data and the final discrimination. The first stage is based on the Lincoln Lab ATR system and the second stage uses some of features (first 10 of the 20 features) used in their system [2,17,18]. In the feature selection stage, we use GA to select a best feature subset, defined as a particular set of features that is the best in discriminating the target from the natural clutter. The diagram for feature selection is given in Fig. 1.

3.1. Feature evaluation

Adding more features does not necessarily improve discrimination performance. An important goal is to choose the best set of features from the discriminating features that are available. Before we do the feature selection, it is appropriate to give a set of feature evaluation criteria, which measure the discrimination capability of each feature or a combination of several features.

3.1.1. Divergence

Divergence is basically a form of the Kulback–Liebler distance measure between density functions. If we assume that the target as well as the natural clutter feature vectors follow the Gaussian distributions, respectively, that is, $N(\mathbf{u}_1, \Sigma_1)$ and $N(\mathbf{u}_2, \Sigma_2)$, where $\mathbf{u}_1$ and $\mathbf{u}_2$ are mean values and Σ_1 and Σ_2 are covariance matrices, respectively, the divergence can be computed as follows

$$d_{12} = \tfrac{1}{2}\mathrm{trace}\{\Sigma_1^{-1}\Sigma_2 + \Sigma_2^{-1}\Sigma_1 - 2I\} + \tfrac{1}{2}(\mathbf{u}_1 - \mathbf{u}_2)^{\mathrm{T}} \times (\Sigma_1^{-1} + \Sigma_2^{-1})(\mathbf{u}_1 - \mathbf{u}_2) \quad (1)$$

One major drawback of the divergence d_{12} is that it is not easily computed, unless the Gaussian assumption is employed. For SAR imagery, the Gaussian assumption itself is in question.

3.1.2. Scatter matrices

These criteria are based upon the information related to the way feature vector samples are scattered in the l-dimensional feature space. We define two kinds of scatter matrices, that is, within-class scatter matrix and between-class scatter matrix. Within-class scatter matrix for M classes is, $S_w = \sum_{i=1}^{M} P_i S_i$, where S_i is the covariance matrix for class ω_i and P_i is the a priori probability of class ω_i. S_w matrix measures how feature vector samples are scattered within each class. Between-class scatter matrix S_b, is defined as follows: $S_b = \sum_{i=1}^{M} P_i(\mathbf{u}_i - \mathbf{u}_0)(\mathbf{u}_i - \mathbf{u}_0)^{\mathrm{T}}$, where $\mathbf{u}_0$ is the global mean vector and $\mathbf{u}_i$ is the mean for each class, $i = 1, \ldots, M$. The between-class scatter matrix measures how the feature vector samples are scattered between different classes. Based on the different combinations of these two scatter matrices, a set of class separability criteria can be derived; one such measure can be defined as: $J = |S_b|/|S_w|$. If the feature vector samples within each class are scattered compactly and the feature vector samples from different classes are far away from one another, we expect the value for J would be high. This also implies that the features we choose have high discrimination.

3.1.3. Feature vector evaluation using a classifier

Another method for feature evaluation depends on the specific classifier. The task of feature selection is to select or determine a set of features, when fed into the classifier, will let the classifier achieve the best performance. So it makes sense to relate the feature selection procedure with the particular classifier used. During the training time, we have all the features extracted from the training data. What we can do is to select a subset of features and feed them into the classifier and see the classification result. Then the goodness of each feature subset is indicated by its classification error rate.

3.2. Various criteria for fitness functions

We use GA to seek the smallest (or the least costly) subset of features for which the classifier's performance does not deteriorate below a certain specified level [9,19]. The basic system framework is shown in Fig. 1.

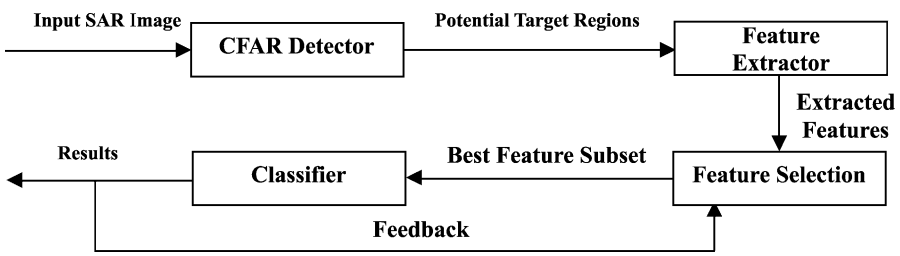


Fig. 1. System diagram.

When the error of a classifier is used to measure the performance, a subset of features is defined as feasible if the classifier's error rate is below the so-called *feasibility threshold*. We search for the smallest subset of features among all feasible subsets. During the search, each subset can be coded as a d-element bit string (d is the total number of features). The ith element of the bit string assumes 0 if the ith feature is excluded from the subset and 1 if it is present in the subset.

In order for the GA to select a subset of features, a fitness function must be defined to evaluate the performance of each subset of features. GA explores the space of subset of features to try to find a minimum subset of features with good classification performance.

3.2.1. Fitness function based on MDLP

In our system, the classifier is fixed, which is a Bayesian classifier, but the set of features that is input into the classifier is a variable. In order to apply MDLP to feature selection, we view the features selected by GA as the model used to describe the training data. Selecting more features means that a more complex model is used to approximate the data. Although a complex model may have perfect performance on the training data, it may not be a good model, since it may be overly sensitive to statistical irregularities and idiosyncrasies of the data and causes accidental noise to be modeled as well, leading to the poor performance on the unseen test data.

To fix the above problem, we use the MDLP to prevent the overfitting of the training data by an overly complex model. Roughly speaking, the MDLP states that among all the models approximating the data to or above certain accuracy, the simplest one is the best one. To restrict the model from growing too complex while maintaining the description accuracy, the cost of describing a set of data with respect to a particular model is defined as the sum of the length of the model and the length of the data when encoded using the model as a predictor for the data. The description length of data-to-model error is defined as the combined length of all data items failed to be described by the model. GA is used to select the subset of features minimizing the above cost. Here, both description lengths are measured in bits and the details of the coding techniques are relevant. The trade-off between simplicity and complexity of both lengths is that if a model is too simple, it may not capture the characteristics of the data and lead to increased error coding length; if a model is too complicated, it may model the noise and become too sensitive to minor irregularities to give accurate prediction of the unseen data. The MDLP states that among the given set of models, the one with the minimum combined description lengths of both the model and data-to-model error is the best approximation and can perform best on the unseen test data.

Based on MDLP, we propose the following fitness function for GA to maximize

$$F(c_i) = -(k \log(f) + n_e \log(n)) \tag{2}$$

where c_i is a chromosome coding the selected set of features, f is the total number of features extracted from each training data, k is the number of features selected (c_i has k bits of 1 and $f - k$ bits of 0), n is the total number of data items in the training set and n_e is the number of data items misclassified. It is easy to see that the fewer the number of features selected and smaller the number of data items misclassified, the larger the value of fitness function.

We now give a brief explanation of the above fitness function. Suppose a sender and a receiver both know all the data items and their order in the training set and also they agree in advance on the feature extractor used to extract the f features from each data item and the classifier used to classify each data based on the features extracted. But only the sender knows the label (target or clutter) of each data item. Now, the sender wants to tell the receiver the label of each data item. One simple approach to do this is to send a bit sequence of n bits where 1 represents the target and 0 represents the clutter. If n is large, then the communication burden will be heavy. In order to reduce the number of bits to be transmitted, in an alternative approach, the sender can tell the receiver which features can be used to classify the data, since the receiver can extract the features and apply the classifier on the features extracted to get the label of each data item. There are a total of f features and $\log(f)$ bits are needed to encode the index of each feature. If k features are selected, $k \log(f)$ bits are needed in order to inform the receiver which features should be extracted. However, some data items may be misclassified, so the sender needs to tell the receiver which data items are misclassified so that the receiver can get the correct labels of all the data in the training set. Since there are a total of n data items, $\log(n)$ bits are needed to encode the index of each data item. If n_e data items are misclassified, then $n_e \log(n)$ bits are needed to convey to the receiver the indices of these misclassified data items. If the set of features selected is effective in discriminating targets from clutter, n_e may be very small, thus the number of bits needs to be transmitted is much smaller than n.

3.2.2. Other fitness functions

We have three other fitness functions to drive GA and compare their performances with that of the fitness function based on MDLP.

In order to define two other fitness functions, we first define the following penalty function [19]

$$p(e) = \frac{\exp((e - t)/m) - 1}{\exp(1) - 1} \tag{3}$$

where e is the error rate (the number of misclassified data item divided by the total number of data items in the training set) of the classifier, t the feasibility threshold and m is

called the 'tolerance margin'. In this paper, $t = 0.01$ and $m = 0.005$. We can see easily that if $e < t$, $p(e)$ is negative and as e approaches zero, $p(e)$ slowly approaches its minimal value. Note also that $p(t) = 0$ and $p(t + m) = 1$. For greater values of the error rate, this penalty function rises quickly toward infinity.

The *second* fitness function is defined as follows

$$F(c_i) = -p(e) \tag{4}$$

This fitness function considers only the error rate of the classifier and does not care about how many features are selected. It can be predicted that this fitness function may lead to the selection of many features.

The *third* fitness function takes the complexity of the model, that is the number of features selected, into consideration. It combines the complexity of the model and its performance on the training data and is defined as follow

$$F(c_i) = -(\gamma \times \text{number\_of\_features} + (1 - \gamma)p(e)) \tag{5}$$

where γ ranges from 0 to 1 and determines the relative importance of the number of features selected and the error rate of the classifier. If we want to use fewer features, we can assign a large value to γ; if we think lower error rate is more important, we can assign a small value to γ. In our experiments, γ takes value 0.1, 0.3, and 0.5.

The *fourth* fitness function is defined as follows

$$F(c_i) = -\left(\gamma \frac{k}{20} + (1 - \gamma)e\right) \tag{6}$$

where k is the number of features selected by GA and γ ranges from 0 to 1 and is a parameter that determines the relative importance of the number of feature selected and the error rate of the classifier.

GA tries to maximize these three fitness functions in order to find an optimal set of features for discriminating targets from clutter.

3.3. System description

3.3.1. CFAR detector

A two-parameter CFAR detector is used as a prescreener to identify potential targets in the image on the basis of radar amplitude. A guard area around a potential target pixel is used for the estimation of clutter statistics. The amplitude of the test pixel is compared with the mean and standard deviation of the clutter according to the following rule

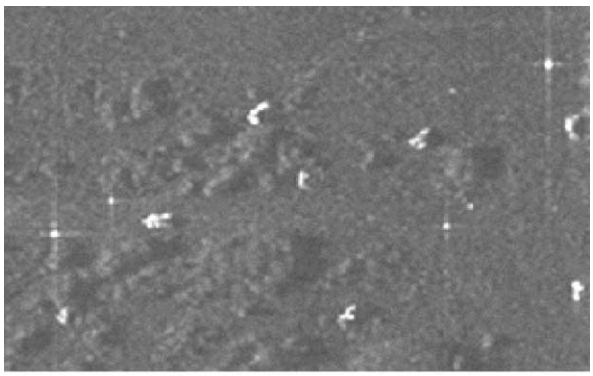
$$\frac{X_t - \hat{u}_c}{\hat{\sigma}_c} > K_{\text{CFAR}} \Rightarrow \text{target, otherwise clutter} \tag{7}$$

where X_t is the amplitude of the test pixel, $\hat{u}_c$ is the estimated mean of the clutter amplitude, $\hat{\sigma}_c$ is the estimated standard deviation of the clutter amplitude, and K_{CFAR} is a constant threshold value that defines the false-alarm rate.

Only those test pixels whose amplitude is much higher than that of the surrounding pixels are declared to be targets. The higher we set the threshold value of K_{CFAR}, the more a test pixel must stand out from its background for it to be declared as a target. Because a single target can produce multiple CFAR detections, the detected pixels are grouped together if they are within a target-sized neighborhood. The CFAR detection threshold in the prescreener is set relatively low to obtain a high initial probability of detection for the target data. It is the responsibility of the discriminator to capture and reject those escaping clutter false alarms from the prescreener stage. An example SAR image and corresponding detection results are shown in Fig. 2.

3.3.2. Feature extractor

First, we use a target-sized rectangular template to determine the position and orientation of the detected target [20]. The algorithm slides and rotates the template until the energy within the template is maximized. Then we extract a set of features from the target-sized template or the region of interest. By using this set of features, we attempt to discriminate the targets from the natural clutter. First ten features are the same as those used in Ref. [2]. All the features from eleven to twenty are not used in their work, they are general features used in pattern recognition and object recognition.



(a)

(b)

Fig. 2. SAR image and CFAR detection result. (a) Example SAR image. (b) Detection result.

The standard-deviation feature (*feature 1*). The standard deviation of the data within the template is a statistical measurement of the fluctuation of the pixel intensities. If we use $P(r,a)$ to represent the radar intensity in power from range r and azimuth a, the standard deviation can be calculated as follows

$$\sigma = \sqrt{\frac{S_2 - \frac{S_1^2}{N}}{N-1}} \quad \text{where}$$
$$S_1 = \sum_{r,a \in \text{region}} 10 \log_{10} P(r,a) \tag{8}$$
$$S_2 = \sum_{r,a \in \text{region}} [10 \log_{10} P(r,a)]^2$$

and N is the number of points in the region.

Targets usually exhibit much larger standard deviation than the natural clutter, as illustrated in Fig. 3.

The fractal dimension feature (*feature 2*). The fractal dimension of the pixels in the region of interest provides information about the spatial distribution of the brightest scatterers of the detected object. It complements the standard-deviation feature, which depends only on the intensities of the scatterers, not on their spatial locations.

The first step in applying the fractal-dimension concept to a radar image is to select an appropriately sized region of interest, and then convert the pixel values in the region of interest to binary. One method of performing this conversion is to select the N brightest pixels in the region of interest and convert their values to 1, while converting the rest of pixel values to 0. Based on these N brightest pixels, we approximate the fractal dimension by using the following formula:

$$\text{dim} = -\frac{\log M_1 - \log M_2}{\log 1 - \log 2} = \frac{\log M_1 - \log M_2}{\log 2} \tag{9}$$

where M_1 represents the minimum number of 1-pixel-by-1-pixel boxes that cover all N brightest pixels in the region of interest (This number is obviously equal to N) and M_2 represents the minimum number of 2-pixel-by-2-pixel boxes required to cover all N brightest pixels.

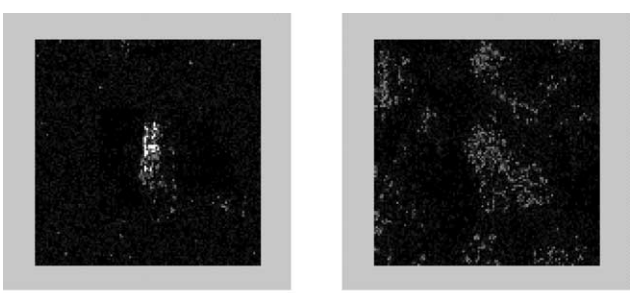


Fig. 3. Example of the standard deviation feature. (a) A typical target image with standard deviation 5.2832. (b) A typical target image with standard deviation 4.5187.

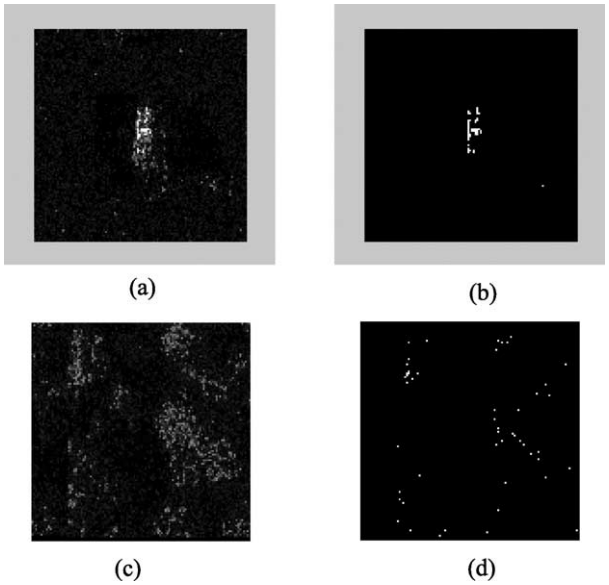


Fig. 4. Example of the fractal dimention feature. (a) Target image. (b) 50 brightest pixels in (a). (c) Natural clutter image. (d) 50 brightest pixel in (c).

The bright pixels for a natural clutter tend to be widely separated, thus produce a low value for the fractal dimension, while the bright pixels for the target tend to be closely bunched, thus we expect a high value for the fractal dimension, which is illustrated in Fig. 4. Fig. 4(a) shows a target image chip. In Fig. 4(b), the 50 brightest pixels from the target image are tightly clustered, and 22 2×2-pixel boxes are needed to cover them, which results in a high fractal dimension of 1.2.

Fig. 4(c) shows a natural clutter image chip. In Fig. 4(d), the 50 brightest pixels from this natural clutter are relatively isolated, and 46 2×2-pixel boxes are needed to cover them, which results in a low fractal dimension of 0.29.

Weighted-rank fill ratio feature (*feature 3*). This textual feature measures the percentage of the total energy contained in the brightest scatterers of a detected object. We define the weighted-rank fill ratio as follows

$$\eta = \frac{\sum_{k \text{ brightest pixels}} P(r,a)}{\sum_{\text{all pixels}} P(r,a)} \tag{10}$$

This feature attempts to exploit the fact that power returns from most targets tend to be concentrated in a few bright scatters, whereas power returns form natural-clutter false alarm tend to be more diffuse. The weighted-rank fill ratio values of target in Fig. 3(a) and clutter in Fig. 3(b) are 0.3861 and 0.2321, respectively.

Size-related feature (*features 4–6*). The three size-related features utilize only the binary image created by the morphological operations on the CFAR detection result.

1. The mass feature is computed by counting the number of pixels in the morphological blob.

2. The diameter is the length of the diagonal of the smallest rectangle that encloses the blob.
3. The square-normalized rotational inertia is the second mechanical moment of the blob around its center of mass, normalized by the inertia of an equal mass square.

In our experiments, we found the size features are not effective in scenarios where the targets are partially occluded or hidden. After the prescreener stage, the size and the shape of the detected morphological blob can be arbitrary. For the clutter, there is also no ground to assert that the resulting morphological blob will exhibit a certain amount of coherence. The experimental results in Fig. 5 show the arbitrariness of the morphological blobs for the targets as well as the clutter.

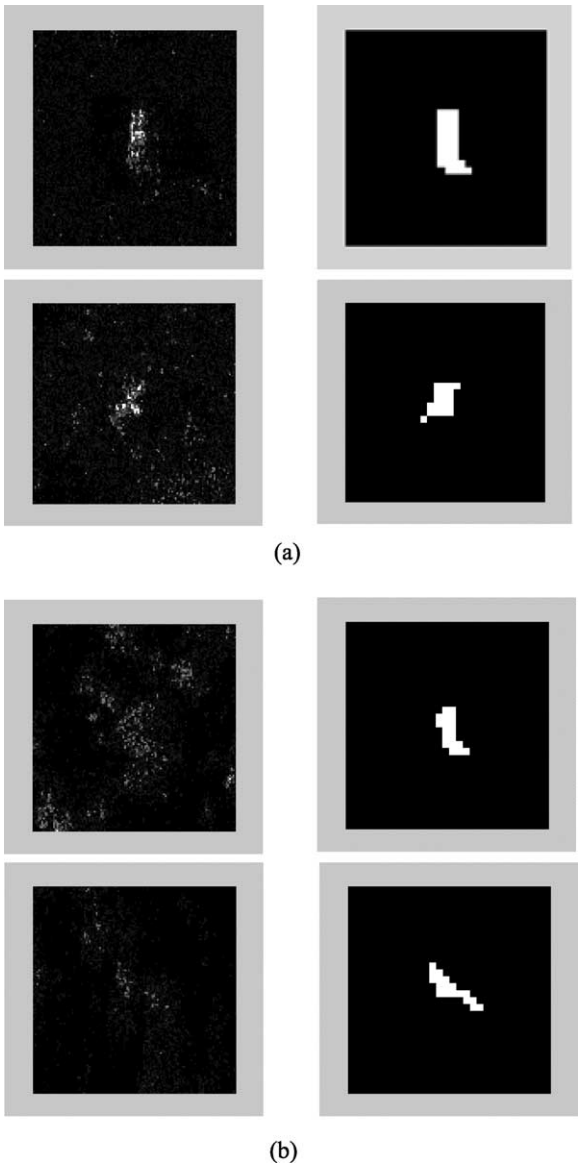


Fig. 5. Example of the size feature for (a) targets and (b) clutter. (a) The lift-hand side figures represent the target images and right-hand side figures represent their corresponding morphological blobs. (b) The lift-hand side figures represent the clutter images and right-hand figures represent their corresponding morphological blobs.

The contrast-based features (*features 7–9*). The CFAR statistic is computed for each pixel in the target-shaped blob to create a CFAR image. Then the three features can be derived as follows:

1. The maximum CFAR feature is the maximum value in the CFAR image contained within the target-sized blob.
2. The mean CFAR feature is the average of the CFAR image taken over the target-shaped blob.
3. The percent bright CFAR feature is the percentage of pixels within the target-sized blob that exceed a certain CFAR value.

The maximum CFAR feature, the mean CFAR feature and the percent bright CFAR feature values of target in Fig. 3(a) are 55.69, 5.53, and 0.15, respectively, and these feature values of clutter in Fig. 3(b) are 10.32, 2.37, and 0.042, respectively. We can see that CFAR feature values for the target are much larger than those for the natural clutter false alarm.

The count feature (*feature 10*). The count feature is very simple; it counts the number of pixels that exceeded the threshold T and normalize this value by the total possible number of pixels in a target blob. The threshold T is set to the quantity corresponding to the 98th percentile of the surrounding clutter. The count feature values of target in Fig. 3(a) and clutter in Fig. 3(b) are 0.6 and 0.1376, respectively. We can see that the count feature value for the target is much larger than that for the natural clutter false alarm. This makes sense because the intensity values of the pixels belonging to the target stand out from the surrounding clutter, while the natural clutter false alarms do not have this property.

The following 10 features, four projection features, three distance features and three moment features, are common features used in image processing and object recognition. They are extracted from the binary image resulting from the CFAR detection. In these images, foreground pixels (pixels with value 1) are potential target pixels.

Projection features (*features 11–14*). four projection features are extracted from each binary image:

1. *horizontal projection feature*: project the foreground pixels on a horizontal line (x-axis of image) and compute the distance between the leftmost point and the rightmost point;
2. *vertical projection feature*: project the foreground pixels on a vertical line (y-axis of image) and compute the distance between the uppermost point and the lowermost point.
3. *major diagonal projection feature*: project the foreground pixels on the major diagonal line and

compute the distance between the upper leftmost point and the lower rightmost point.
4. *minor diagonal projection feature*. project the foreground pixels on the minor diagonal line and compute the distance between the lower leftmost point and the upper rightmost point.

The average values of horizontal, vertical, major and minor diagonal projection features of all the clutter images, we collected, are approximately 60.0, 60.0, 90.0, and 90.0, respectively. Their corresponding values for target images are 34.5, 29.5, 46.7, and 47.8, respectively. It can be seen that the feature values for the clutter are larger than those for the target. This result is reasonable, since the bright pixels of a natural clutter tend to be widely separated. This has already been shown by the fractal dimension feature value.

Distance features (*features 15–17*). Three distance features are extracted from each binary image. Before computing distance features, we first compute the centroid of all the foreground pixels in the binary image.

1. *minimum distance*: compute the distance from each foreground pixel to the centroid and select the minimum one.
2. *maximum distance*: compute the distance from each foreground pixel to the centroid and select the maximum one.
3. *average distance*: compute the distance from each foreground pixel to the centroid and get the average value of all these distances.

The average values of minimum, maximum and average distance features of all the clutter images we collected are approximately 40.0, 70.0, and 60.0, respectively. Their corresponding values of target images are 3.8, 26.7, and 11.5, respectively. It can be seen that the feature values for the clutter are larger than those for the target. This result is reasonable, since the bright pixels of a natural clutter tend to be widely separated.

Moment features (*features 18–20*). Three moment features are extracted from each binary image. All three moments are central moments, so before computing moment features, we first compute the centroid of all the foreground pixels in the binary image.

The central moments can be expressed as

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q \mathrm{d}x\, \mathrm{d}y \tag{11}$$

where $(\bar{x}, \bar{y})$ is the centroid and p and q are integers.

We compute μ_{20}, μ_{02} and μ_{22} from each binary image and we call them horizontal, vertical and diagonal second-order moment features, respectively.

The average values of horizontal, vertical and diagonal second-order moment features of all the clutter images we collected are approximately 910.0, 910.0, and 37,4020.0, respectively. Their corresponding values of target images are 80.5, 46.7, and 4021.6, respectively. It can be seen that the feature values for the clutter are larger than those for the target. This result is reasonable, since the bright pixels of a natural clutter tend to be widely separated.

3.3.3. GAs for feature selection

The GA is an optimization procedure that operates in binary search spaces (the search space consists of binary strings). A point in the search space is represented by a finite sequence of 0s and 1s, called a *chromosome*. The algorithm manipulates a finite set of chromosomes, the *population*, in a manner resembling the mechanism of natural evolution. Each chromosome is evaluated to determine its 'fitness', which determines how likely the chromosome is to survive and breed into the next generation. The probability of survival is proportional to the chromosome's fitness value. Those chromosomes which have higher fitness values are given more chances to 'reproduce' by the processes of *crossover* and *mutation*. The function of crossover is to mate two parental chromosomes to produce a pair of offspring chromosomes. In particular, if a chromosome is represented by a binary string, crossover can be implemented by randomly choosing a point, called the crossover point, at which two chromosomes exchange their parts to create two new chromosomes. Mutation randomly perturbs the bits of a single parent to create a child. This procedure can increase the diversity of the population. Mutations can be performed by flipping randomly one or more bits in chromosomes. In this paper, we implement an adaptive GA that can automatically adapt the parameters such as crossover rate and mutation rate based on the performance of GA. To be specific, if the fitness value of the best individual is not improved for three or five generations in a row, GA will automatically raise the mutation rate to increase the diversity of the population. Also, elitism mechanism is adopted such that the best individual (set of features selected) is copied from generation to generation when performing reproduction.

In this paper, there are 20 features as described earlier. Each feature is represented as a bit in the GA. There are 2^{20} possible combinations of these features.

4. Experimental results

We use SAR images from MSTAR public data (target and clutter data) and generate 1008 target chips (small SAR images containing target) and 1008 clutter chips (small SAR images containing clutter) of size 120×120. We also use SAR images that are downloaded from the website of MIT Lincoln Lab. From these SAR images, 40 target chips and 40 clutter chips of size 120×120 are generated. By adding these two sets of images, we have 1048 target chips and 1048 clutter chips. Some of the chips are used in training and the rest are used in testing. The chips used in training are

randomly selected. The GA selects a good subset of features from the 20 features described previously to classify a SAR image chip into either a target or a clutter. We use the CFAR detector in the prescreener stage to detect the potential target regions. Since we know the ground truth, we know which one is the real target and which one is the clutter false alarm among the potential target regions detected. This allows us to construct a set of training data (training target data and training natural clutter false alarm data) for feature selection. Then we extract a set of 20 features from each potential target region and do the feature selection. Finally in the testing stage we use the selected features to discriminate the targets from the natural clutter false alarms.

For our GA-based feature selection framework, we adopt a Bayesian Classifier to classify the training data and the resulting error rate is used as the feedback into the feature selection algorithm. The size of the population is 100, the initial crossover rate is 0.8 and the initial mutation rate is 0.01. If the fitness value of the best individual is not improved for three generations in a row, GA increases the mutation rate by 0.02. In order to reduce the training time, we set an error rate threshold ε. The GA stops when either the error rate of the best set of features selected is below the specified threshold ε or the mutation rate is increased above 0.09.

We carried out a series of experiments to test the efficacy of GA in feature selection. First, we use the MDLP-based fitness function. Then we use the other three fitness functions. Finally, we compare and analyze the performances of these fitness functions. In order to have an objective comparison of various experiments, the GA is invoked 10 times for each experiment with the same set of parameters and the same set of training chips. Only the average performances are used for comparison.

4.1. MDLP-based fitness function

We performed four experiments with this fitness function. In the first experiment, 300 target chips and 300 clutter chips are used in training and 748 target chips and 748 clutter chips are used in testing, the error rate threshold value ε is 0.002; in the second experiment, 500 target chips and 500 clutter chips are used in training and 548 target chips and 548 clutter chips are used in testing, the error rate threshold value ε is 0.0015; in the third and fourth experiments, 700 target chips and 700 clutter chips are used in training and 348 target chips and 348 clutter chips are used in testing, the error rate threshold value ε is 0.0015 and 0.0011, respectively. The features selected during training are used for classification during testing. It is worth noting that the training chip set in the third and fourth experiments is the superset of that in the second experiment and the training chip set in the second experiment is the superset of that in the first experiment. The target and clutter chips used during training are selected at random.

Table 1 shows the experimental results where 300 target and 300 clutter chips are used in training. GA is invoked 10 times and each row records the experimental results from the corresponding invocation. The last row records

Table 1
Experimental results with 300 training target and clutter chips (MDLP, Eq. (2); $\varepsilon = 0.002$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	29	47	4	0100101001 0000000000	0.003	1	1	0.001	0	2
2	9	27	6	0110001011 0000100000	0.003	1	1	0.011	0	16
3	10	28	4	0100001001 0100000000	0.003	1	1	0.011	0	16
4	43	61	4	0000001001 0100100000	0.003	1	1	0.005	0	7
5	19	37	4	0101001001 0000000000	0.003	1	1	0.017	0	25
6	13	31	4	0100001001 1000000000	0.003	1	1	0.007	0	10
7	23	41	4	0100001001 0010000000	0.003	1	1	0.011	0	16
8	6	24	6	0010011011 0000100000	0.003	1	1	0.011	0	16
9	17	35	5	0100001001 0011000000	0.003	1	1	0.003	0	5
10	11	29	5	0100001001 0010100000	0.003	1	1	0.005	0	7
Ave	18	36	4.6		0.003	1	1	0.0082	0	12

Table 2
Experimental results with 500 training target and clutter chips (MDLP, Eq. (2); $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	17	35	5	0100001001 1000100000	0.002	1	1	0.006	0	7
2	13	31	5	0100001001 0000001001	0.002	1	1	0.006	0	7
3	19	38	5	0100001001 0000011000	0.002	1	1	0.006	0	7
4	20	38	5	0100001001 0000011000	0.002	1	1	0.006	0	7
5	10	28	5	0100001001 0010100000	0.002	1	1	0.006	0	7
6	26	44	5	0100001001 1100000000	0.002	1	1	0.003	0	3
7	25	43	5	0100001001 0000010100	0.002	1	1	0.007	0	8
8	9	27	6	0000001011 0000011010	0.002	1	1	0.007	0	8
9	8	26	5	0100001001 0000011000	0.002	1	1	0.006	0	7
10	17	35	5	0001001001 0011000000	0.002	1	1	0.004	0	4
Ave	16.4	34.5	5.1		0.002	1	1	0.0057	0	6.5

the average results of 10 runs. The column 'Best generation' records the generation number in which the best set of features is found and the column 'Total generation' shows the total number of generations GA runs. It can be seen that although the training error rate is 0.003 in each run, different features are selected. In some runs, the same number of testing clutter chips are misclassified, but the clutter chips that are misclassified in each run are different. From the testing results, we can observe that sometimes clutter chips are misclassified as target chips. The testing results show that GA finds an effective set of features to discriminate target from clutter.

Table 3
Experimental results with 700 training target and clutter chips (MDLP, Eq. (2); $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	8	8	9	0101101001 1010001001	0.0014	1	1	0.006	0	4
2	9	9	10	1101001001 1010101010	0.0014	1	1	0.001	0	1
3	7	7	7	0000001011 0100101010	0.0014	1	1	0.012	0	8
4	2	2	10	1101001001 0110011010	0.0014	1	1	0.001	0	1
5	5	5	8	0100001001 0011111000	0.0014	1	1	0.007	0	5
6	2	2	7	1000011011 0100001000	0.0014	1	1	0.012	0	8
7	5	5	10	1101001001 0110101100	0.0014	1	1	0.001	0	1
8	3	3	10	1100101011 0101010001	0.0014	1	1	0.003	0	2
9	4	4	11	1101011001 1010111000	0.0014	1	1	0.001	0	1
10	4	4	10	1101001001 0011111000	0.0014	1	1	0.001	0	1
Ave	4.9	4.9	9.2		0.0014	1	1	0.0045	0	3.2

Table 4
Experimental results with 700 training target and clutter chips (MDLP, Eq. (2); $\varepsilon = 0.0011$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	10	28	6	0001001001 1000001010	0.0014	1	1	0.004	0	3
2	19	37	5	0100001001 0000001010	0.0014	1	1	0.012	0	8
3	17	35	5	0100001001 0010100000	0.0014	1	1	0.01	0	7
4	16	34	6	0001011001 0010001000	0.0014	1	1	0.006	0	4
5	16	34	5	0100001001 0000011000	0.0014	1	1	0.01	0	7
6	19	37	5	0100001001 0010100000	0.0014	1	1	0.01	0	7
7	10	28	5	0100001001 0000010100	0.0014	1	1	0.01	0	7
8	15	33	5	0100001001 0000011000	0.0014	1	1	0.01	0	7
9	10	28	6	0100011001 1000010000	0.0014	1	1	0.007	0	5
10	23	41	5	0100001001 0000001001	0.0014	1	1	0.01	0	7
Ave	15.5	33.5	5.3		0.0014	1	1	0.0089	0	6.1

Tables 2 and 3 show the experimental results when 500 target and clutter chips and 700 target and clutter chips are used in training, respectively. The results in Table 2 are very good. On the average, 5.1 features are selected and both the training and testing error rate are very low. However, the results in Table 3 are not good. Although the training and testing error rates are low, 9.2 features are selected on the average. From Table 3, we can see that GA runs 4.9 generations on the average. It is clear that GA stops prematurely. The reason for the premature termination is that the error rate threshold value 0.0015 is high in this case, since there are 700 target chips and 700 clutter chips. In order to force GA to explore the search space, we lower the error rate threshold value to 0.0011 and get the results shown in Table 4. These results are much better than those in Table 3. Only 5.3 features are selected on the average, although the average testing error rate is almost doubled. Considering both the test error rate and the number of features selected, the first run in Tables 1 and 4, and the sixth run in Table 2 yield the best results. Fig. 6 shows how fitness values change as GA searches the feature subset space during these runs; Fig. 7 shows how training error rate changes and Fig. 8 shows how the number of features selected changes.

From the above experiments, we can see that the MDLP-based fitness function and adaptive GA are very efficient in feature selection. Only 4–6 features are selected on the average while the detection accuracy is kept high.

4.2. Other fitness functions

For the purpose of objective comparison, the training chip set in the following experiments is the same as that in the second experiment above, that is, 500 target chips and 500 clutter chips are used in training and 548 target chips and 548 clutter chips are used in testing.

First, we use function (4) as the fitness function and invoke GA 10 times. The error rate threshold value is

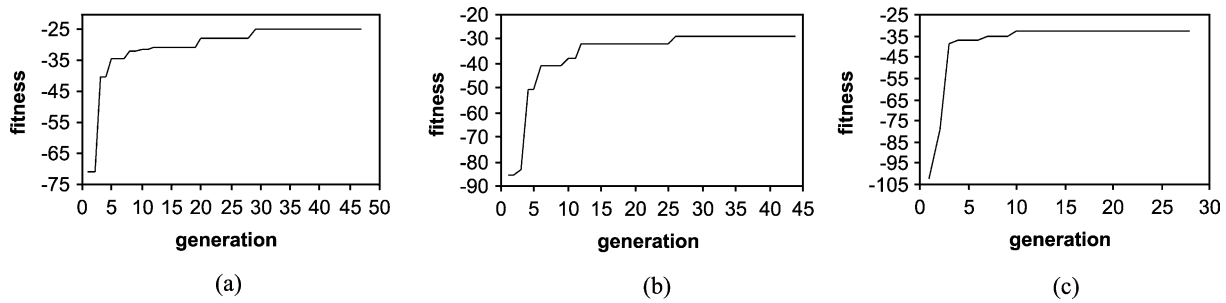


Fig. 6. Fitness values vs. generation number. (a) 300 training target and clutter chips. (b) 500 training target and clutter chips. (c) 700 training target and clutter chips.

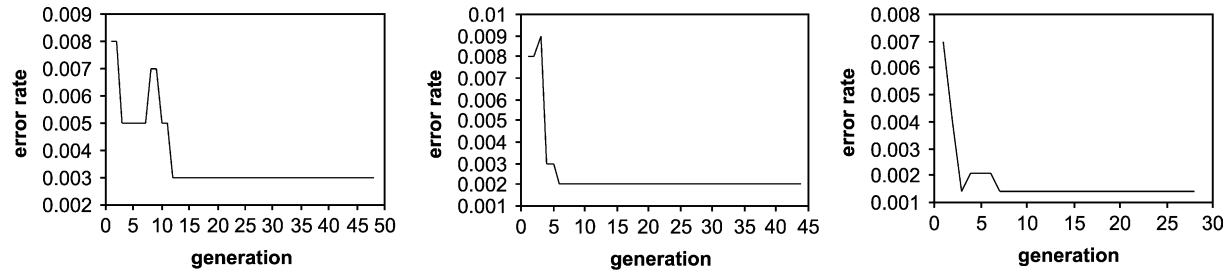


Fig. 7. Error rates vs. generation number. (a) 300 training target and clutter chips. (b) 500 training target and clutter chips. (c) 700 training target and clutter chips.

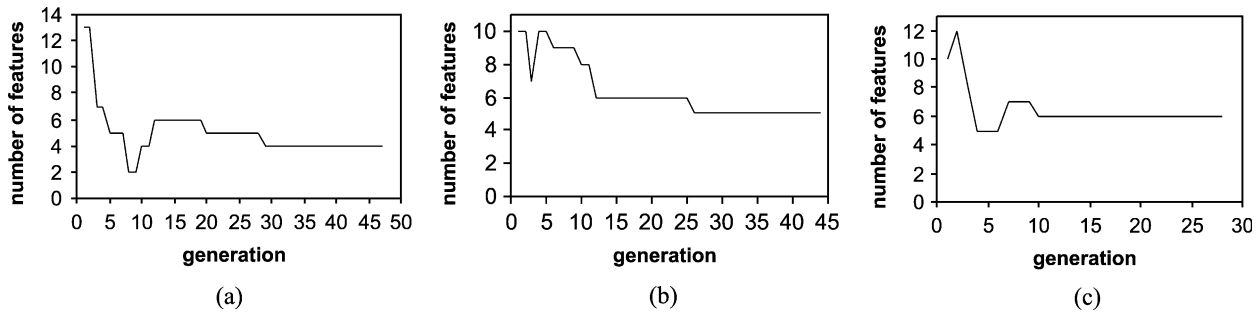


Fig. 8. The number of features selected vs. generation number. (a) 300 training target and clutter chips. (b) 500 training target and clutter chips. (c) 700 training target and clutter chips.

0.0015. Table 5 shows the experimental results. This function is only dependent on the error rate, so GA found a set of features with very low error rate quickly. The selected features are shown by the 'Number of features' and 'Features selected' columns. However, since the number of features is not taken into consideration by the fitness function, many features are selected. More than 10 features are selected on the average in 10 runs.

Next, we use function (5) as the fitness function. We performed three experiments with this function, and

Table 5
Experimental results with 500 training target and clutter chips (penalty function, Eq. (4); $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	4	22	13	0111111011 1100111000	0.002	1	1	0.004	0	4
2	11	11	10	1011011011 0001100100	0.001	1	0	0.005	0	5
3	2	20	9	0101101001 1011000100	0.002	1	1	0.005	0	5
4	4	22	11	1010011011 0101011100	0.002	1	1	0.004	0	4
5	3	21	10	1110001011 1010010100	0.002	1	1	0.003	0	3
6	10	10	9	0011011011 0000110100	0.001	1	0	0.005	0	5
7	8	26	10	1101101001 0011010010	0.002	1	1	0.001	0	1
8	2	20	11	1110101011 0001001110	0.002	1	1	0.003	0	3
9	3	21	10	0110011011 1101100000	0.002	1	1	0.005	0	5
10	3	21	9	1110011011 0000110000	0.002	1	1	0.008	0	9
Ave	5	19.4	10.2		0.0018	1	1	0.0043	0	4.4

Table 6
Experimental results with 500 training target and clutter chips (penalty and number of features, Eq. (5); $\gamma = 0.1$; $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	18	36	2	0000001001 0000000000	0.005	2	3	0.024	0	26
2	12	30	2	0000001000 0000001000	0.007	1	6	0.007	0	8
3	17	35	2	0000001001 0000000000	0.005	2	3	0.024	0	26
4	20	38	2	0000001001 0000000000	0.005	2	3	0.024	0	26
5	16	34	2	0000001001 0000000000	0.005	2	3	0.024	0	26
6	11	29	2	0000001001 0000000000	0.005	2	3	0.024	0	26
7	15	33	2	0000001001 0000000000	0.005	2	3	0.024	0	26
8	17	35	2	0000001001 0000000000	0.005	2	3	0.024	0	26
9	14	32	2	0000001001 0000000000	0.005	2	3	0.024	0	26
10	12	30	2	0000001000 0000001000	0.007	1	6	0.007	0	8
Ave	15.2	33.2	2		0.0054	1.8	3.6	0.0206	0	22.4

the values of γ are 0.1, 0.3, and 0.5 in these three experiments, respectively. The error rate threshold is 0.0015. Since this function considers the number of features selected, you can imagine that few features will be selected. Tables 6–8 show the corresponding experimental results when γ is 0.1, 0.3, and 0.5.

From Table 6, we can see that since the training error rate is low, the number of features selected accounts for a large percentage of the value of the fitness function, forcing GA to select only two features in each run. However, the error rate for testing results is not encouraging. It is more than 0.02 on the average.

Table 7
Experimental results with 500 training target and clutter chips (penalty and number of features, Eq. (5); $\gamma = 0.3$; $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	23	41	1	0000001000 0000000000	0.01	1	9	0.036	0	39
2	20	38	1	0000001000 0000000000	0.01	1	9	0.036	0	39
3	11	29	2	1000001000 0000000000	0.005	1	4	0.033	0	36
4	8	26	3	0000000010 0010010000	0.008	4	4	0.005	0	5
5	30	48	1	0000001000 0000000000	0.01	1	9	0.036	0	39
6	14	32	1	0000001000 0000000000	0.01	1	9	0.036	0	39
7	25	43	1	0000001000 0000000000	0.01	1	9	0.036	0	39
8	20	38	1	0000001000 0000000000	0.01	1	9	0.036	0	39
9	22	40	1	0000001000 0000000000	0.01	1	9	0.036	0	39
10	27	45	1	0000001000 0000000000	0.01	1	9	0.036	0	39
Ave	20	38	1.3		0.0093	1.3	8	0.0326	0	35.3

Table 8
Experimental results with 500 training target and clutter chips (penalty and number of features, Eq. (5); $\gamma = 0.5$; $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	17	35	1	0000001000 0000000000	0.01	1	9	0.036	0	39
2	29	41	1	0000001000 0000000000	0.01	1	9	0.036	0	39
3	22	40	1	0000001000 0000000000	0.01	1	9	0.036	0	39
4	15	33	1	0000001000 0000000000	0.01	1	9	0.036	0	39
5	32	50	1	0000001000 0000000000	0.01	1	9	0.036	0	39
6	11	29	1	0000001000 0000000000	0.01	1	9	0.036	0	39
7	11	29	1	0000001000 0000000000	0.01	1	9	0.036	0	39
8	23	41	1	0000001000 0000000000	0.01	1	9	0.036	0	39
9	9	27	2	0000000010 0000001000	0.012	5	7	0.011	0	12
10	23	41	1	0000001000 0000000000	0.01	1	9	0.036	0	39
Ave	19.2	37.2	1.1		0.01	1.5	8.8	0.0335	0	36.3

When γ is 0.3, the number of features account for a larger part of the value of the fitness function than when γ is 0.1, forcing GA to select almost only one feature. Actually, in eight runs, GA selects the best feature among all the 20 features (Table 12) to discriminate the target from clutter. When γ is 0.5, the number of features almost dominates the value of fitness function. The same phenomenon occurs and the experimental results are shown in Table 8.

Finally, we use function (6) as the fitness function. We did three experiments with this function, and the values of γ are 0.1, 0.3, and 0.5 in these three experiments, respectively. The error rate threshold is

Table 9
Experimental results with 500 training target and clutter chips (error rate and number of features, Eq. (6); $\gamma = 0.1$; $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	21	39	1	0000001000 0000000000	0.01	1	9	0.036	0	39
2	16	34	1	0000001000 0000000000	0.01	1	9	0.036	0	39
3	14	32	2	0000100010 0000000000	0.01	7	3	0.006	0	7
4	25	43	1	0000001000 0000000000	0.01	1	9	0.036	0	39
5	13	31	1	0000001000 0000000000	0.01	1	9	0.036	0	39
6	17	35	1	0000001000 0000000000	0.01	1	9	0.036	0	39
7	17	35	2	0000100010 0000000000	0.01	7	3	0.006	0	7
8	33	51	1	0000001000 0000000000	0.01	1	9	0.036	0	39
9	22	40	1	0000001000 0000000000	0.01	1	9	0.036	0	39
10	12	30	1	0000001000 0000000000	0.01	1	9	0.036	0	39
Ave	19	37	1.2		0.01	2.2	7.8	0.03	0	32.6

Table 10
Experimental results with 500 training target and clutter chips (penalty and number of features, Eq. (6); $\gamma = 0.3$; $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	11	29	1	0000001000 0000000000	0.01	1	9	0.036	0	39
2	27	45	1	0000001000 0000000000	0.01	1	9	0.036	0	39
3	17	35	1	0000001000 0000000000	0.01	1	9	0.036	0	39
4	11	29	1	0000001000 0000000000	0.01	1	9	0.036	0	39
5	11	29	1	0000001000 0000000000	0.01	1	9	0.036	0	39
6	11	29	1	0000001000 0000000000	0.019	7	12	0.028	0	31
7	20	38	1	0000000010 0000000000	0.01	1	9	0.036	0	39
8	30	48	1	0000000010 0000000000	0.019	7	12	0.028	0	31
9	7	25	1	0000000010 0000000000	0.019	7	12	0.028	0	31
10	12	30	1	0000001000 0000000000	0.01	1	9	0.036	0	39
Ave	15.7	33.7	1		0.013	2.8	9.9	0.0336	0	36.3

0.0015. Like the function (6), this function considers both the number of features selected and the error rate. When γ is large, this function forces GA to select one feature. Usually, the best feature is selected (Table 12). Tables 9–11 show the corresponding experimental results when γ is 0.1, 0.3, and 0.5, respectively.

In order to show that GA selects the best feature when the number of features dominates the fitness function,

Table 11
Experimental results with 500 training target and clutter chips (penalty and number of features, Eq. (6); $\gamma = 0.5$; $\varepsilon = 0.0015$)

Runs	Best generation	Total generation	Number of features	Features selected	Training error rate	Number of errors		Testing error rate	Number of errors	
						Target	Clutter		Target	Clutter
1	25	43	1	0000000010 0000000000	0.019	7	12	0.028	0	31
2	11	29	1	0000001000 0000000000	0.01	1	9	0.036	0	39
3	8	26	1	0000000010 0000000000	0.019	7	12	0.028	0	31
4	11	29	1	0000001000 0000000000	0.01	1	9	0.036	0	39
5	8	26	1	0000001000 0000000000	0.01	1	9	0.036	0	39
6	15	33	1	0000001000 0000000000	0.01	1	9	0.036	0	39
7	9	27	1	0000001000 0000000000	0.01	1	9	0.036	0	39
8	12	30	1	0000001000 0000000000	0.01	1	9	0.036	0	39
9	29	47	1	0000001000 0000000000	0.01	1	9	0.036	0	39
10	24	42	1	0000001000 0000000000	0.01	1	9	0.036	0	39
Ave	15.2	33.2	1		0.013	2.2	9.4	0.0344	0	37.4

Table 12
Experimental results using only one feature for discrimination (target chips = 500, clutter chips = 500)

Feature	Error rate	Number of errors		Feature	Error rate	Number of errors	
		Target	Clutter			Target	Clutter
1	0.119	17	102	11	0.118	18	100
2	0.099	16	83	12	0.111	6	105
3	0.056	7	49	13	0.126	9	117
4	0.057	17	40	14	0.131	7	124
5	0.068	13	55	15	0.09	5	85
6	0.354	0	354	16	0.069	3	66
7	0.01	1	9	17	0.075	3	72
8	0.5	480	20	18	0.209	0	209
9	0.019	7	12	19	0.2	2	198
10	0.073	15	58	20	0.244	0	244

we examine the efficacy of each feature in discriminating the targets from clutter. The data used in examination are 500 target chips and 500 clutter chips used in the above training. The results are shown in Table 12. From this table, it can be seen that the best feature (feature 7, the maximum CFAR feature) is selected by GA.

4.3. Comparison and analysis

Fig. 9 shows the average performances of the above experiments pictorially. The x-axis is the average number of features selected and the y-axis is the average training error rate. We use the average number of features selected and average training error rate to form a performance point and evaluate the performance according to the location of performance point. A good performance point should have lower values of both the average number of features and the training error. The three points (shown as circles) are the performance points when the MDLP-based fitness function is used and the rest are the performance points corresponding to other fitness functions.

From the above experimental results, we can see that GA is capable of selecting a good set of features to discriminate the target from clutter. The MDLP-based fitness function is the best fitness function compared to three other functions. Fitness function (4) doesn't include the number of features. Although GA can find a good set of features quickly driven by this function, many features are selected. This greatly increases the computational complexity in the testing phase. Fitness functions (5) and (6) take the number of features selected into consideration. However, the number of features dominates the fitness function value, forcing GA to select only one or two features, leading to the unsatisfactory training and testing error rates. In order to balance the number of features selected and the error rate, parameter γ must be finely tuned. This is not an easy task and it usually takes a lot of time. The MDLP-based fitness function is based on a sound theory and it balances these two terms very well. Only a few features are selected while the training and testing error rates are kept low.

In order to evaluate which features are more important than others using MDLP-based approach, we combine the results of first, second and fourth experiments. Note that in the first, second and fourth experiments (Tables 1, 2 and 4), GA is invoked for a total of 30 times. Table 13 shows the number of times each feature is selected in these 30 runs. It can be seen from Table 13 that the fractal dimension feature (feature 2), the maximum CFAR feature (feature 7) and the count feature (feature 10) are very useful in detecting targets in SAR images, while the standard deviation feature (feature 1) and the mean CFAR feature (feature 8) are not good. The major diagonal projection feature (feature 13), the minimum distance feature (feature 15), the maximum distance feature (feature 16) and the average distance feature (feature 17) have low utility while other features have very low utility. These results are consistent with those shown in Table 12. Considered individually, the maximum CFAR feature (feature 7) is

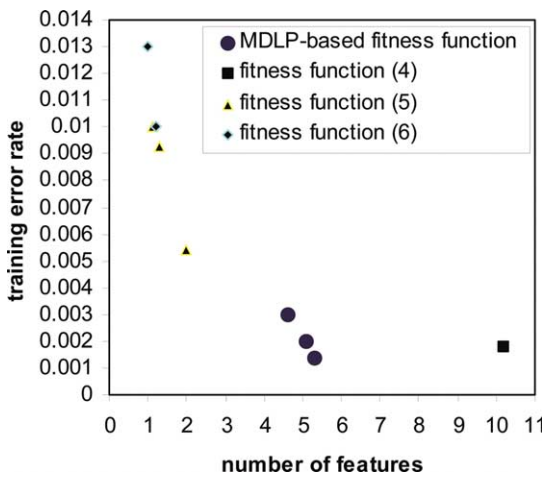


Fig. 9. Average performances of various fitness functions.

Table 13
The number of times each feature is selected in Experiments 1, 2 and 4

	Features																			
	1	2*	3	4	5	6	7*	8	9	10*	11	12	13	14	15	16	17	18	19	20
Exp1	0	8	2	1	1	1	10	0	2	10	1	2	3	1	4	0	0	0	0	0
Exp2	0	8	0	1	0	0	10	0	1	10	2	1	2	1	2	4	5	1	1	1
Exp4	0	8	0	2	0	2	10	0	0	10	1	0	3	0	2	4	6	1	2	1
Total	0	24	2	4	1	3	30	0	3	30	4	3	8	2	8	8	11	2	3	2

the best feature (Table 12) and it is selected by GA (in combination with other features) in all the 30 runs.

5. Conclusion

In this paper, we introduced the GA feature selection algorithm into a specific application domain to discriminate the targets from the natural clutter false alarms in SAR images. Rough target detection, feature extraction, GA feature selection and final discrimination are successfully implemented and good results are obtained. Our experimental results show that the GA selected a good subset of features. Also, we proposed a MDLP-based fitness function and compared its performance with three other fitness functions. Our experimental results show that it balances the number of features selected and the error rate very well and it is the best fitness function compared to other three functions. In the future, we plan to extend this approach to additional features and more complex background clutter.

Acknowledgements

This research is supported by the grant F33615-99-C-1440. The contents of the information do not necessarily reflect the position or policy of the US Government.

References

[1] B. Bhanu, D. Dudgeon, E. Zelnio, A. Rosenfeld, D. Casaseut, I. Reed (Eds). Special Issue on Automatic Target Recognition, IEEE Transactions on Image Processing 6 (1) (1997).
[2] D. Kreithen, S. Halversen, G. Owirka, Discriminating targets from clutter, Lincoln Laboratory Journal 6 (1) (1993) 25–52.
[3] S. Cagnoni, A. Dobrzeniecki, R. Poli, J. Yanch, Genetic algorithm-based interactive segmentation of 3D medical images, Image and Vision Computing 17 (12) (1999) 881–895.
[4] B. Bhanu, T. Poggio, (Eds) Special section on Machine Learning in Computer Vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (9) (1994).
[5] M. Rizki, L. Tamburino, M. Zmuda, Multi-resolution feature extraction from gabor filtered images, Proceedings of the IEEE National Aerospace and Electronics Conference, Dayton, OH, USA, May 1993, pp. 24–28
[6] A. Katz, P. Thrift, Generating image filters for target recognition by genetic learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (9) (1994).
[7] E. Ozcan, C. Mohan, Partial shape matching using genetic algorithms, Pattern Recognition Letters 18 (1997) 987–992.
[8] R. Srikanth, R. George, N. Warsi, D. Prabhu, F. Petry, B. Buckles, A variable-length genetic algorithm for clustering and classification, Pattern Recognition Letters 16 (1995) 789–800.
[9] W. Punch, E. Goodman, Further research on feature selection and classification using genetic algorithms, Proceedings of the Fifth International Conference on Genetic Algorithms (1993) 557–564.
[10] B. Bhanu, S. Lee, Genetic Learning for Adaptive Image Segmentation, Kluwer Academic Publishers, Dordrecht, 1994.
[11] C. Emmanouilidis, A. Hunter, J. MacIntyre, C. Cox, Multiple-criteria genetic algorithms for feature selection in neuro-fuzzy modeling, Proceedings of the International Joint Conference on Neural Networks, Piscataway, NJ, USA, vol. 6, 1999, pp. 4387–4392
[12] P. Estevez, R. Caballero, A niching genetic algorithm for selecting features for neural classifiers, Proceedings of the Eighth International Conference on Artificial Neural Networks, vol. 1, Springer, London, 1998, pp. 311–316.
[13] F. Rhee, Y. Lee, Unsupervised feature selection using a fuzzy-genetic algorithm, Proceedings of the IEEE International Fuzzy Systems Conference, Piscataway, NJ, vol. 3, 1999, pp. 1266–1269
[14] K. Matsui, Y. Suganami, Y. Kosugi, Feature selection by genetic algorithm for MRI segmentation, Systems and Computers in Japan 30 (7) (1999) 69–78. Scripta technical.
[15] J. Quinlan, R. Rivest, Inferring decision tree using the minimum description length principle, Information and Computation 80 (1989) 227–248.
[16] Q. Gao, M. Li, P. Vitanyi, Applying MDL to learn best model granularity, Artificial Intelligence 121 (2000) 1–29.
[17] L. Novak, G. Owirka, C. Netishen, Performance of a high-resolution polarimetric SAR automatic target recognition system, Lincoln Laboratory Journal 6 (1) (1993) 11–24.
[18] L. Novak, M. Burl, W. Irving, Optimal polarimetric processing for enhanced target detection, IEEE Transactions on Aerospace and Electronics Systems 29 (1993) 234–244.
[19] W. Siedlecki, J. Sklansky, A note on genetic algorithms for large-scale feature selection, Pattern Recognition Letters 10 (1989) 335–347.
[20] S. Halversen, Calculating the orientation of a rectangular target in SAR imagery, Proceedings of the IEEE National Aerospace and Electronics Conference (1992) 260–264.

[21] J. Rissanen, A universal prior for integers and estimation by minimum description length, Annals of Statistics 11 (2) (1983) 416–431.

Bir Bhanu received the SM and EE degrees in electrical engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, the PhD degree in electrical engineering from the Image Processing Institute, University of Southern California, Los Angeles. Currently he is a Professor of electrical engineering and computer science and Director of Center for Research in Intelligent Systems at the University of California, Riverside. Previously, he was a Senior Honeywell Fellow at Honeywell Systems and Research Center, Minneapolis, MN. He has been the principal investigator of various programs for DARPA, NASA, NSF, AFOSR, ARO and other agencies and industries in the areas of learning and vision, image understanding, pattern recognition, target recognition, navigation, image databases, and machine vision applications. He is the co-author of books on 'Computational Learning for Adaptive Computer Vision', (Kluwer, Forthcoming), 'Genetic Learning for Adaptive Image Segmentation' (Kluwer, 1994), and 'Qualitative Motion Understanding' (Kluwer, 1992). He holds 10 US and international patents and over 200 reviewed technical publications in the areas of his interest. Dr Bhanu is a Fellow of the IEEE, the IAPR and the AAAS. He is a member of ACM, AAAI, Sigma Xi and SPIE.

Yingqiang Lin received his BS and MS degree in computer science from Fudan University, Shanghai, China in 1991 and 1994, respectively. Currently, he is a computer science PhD candidate at Center for Research in Intelligent System, University of California, Riverside. His research interests include image processing, pattern recognition and Machine Learning.