Performing Classification with an Environment Manipulating Mutable Automata (EMMA)

Karl Benson
Defence Evaluation and Research Agency,
DERA Malvern, St Andrews Road,
Worcestershire WR14 3PS, UK
Tel: +44(0)1684 894580
kabenson@dera.gov.uk

Abstract- In this paper a novel approach to performing classification is presented. Hypersurface Discriminant functions are evolved using Genetic Programming. These discriminant functions reside in the states of a Finite State Automata, which has the ability to reason[1] and logically combine the hypersurfaces to generate a complex decision space. An object may be classified by one or many of the discriminant functions, this is decided by the automata. During the evolution of this symbiotic architecture, feature selection for each of the discriminant functions is achieved implicitly, a task which is normally performed before a classification algorithm is trained. Since each discriminant function has different features, and objects may be classified with one or more discriminant functions, no two objects from the same class need be classified using the same features. Instead, the most appropriate features for a given object are used.

1 Introduction

This paper outlines current research being carried out at DERA Malvern into the evolution of Automatic Target Detection (ATD) algorithms. Evolved algorithms are incorporated into an automatic all source image interpretation system which draws the attention of photographic interpreters to regions of interest within an image. One such algorithm known as the FSM(GP) was developed by Benson [2] for the automatic detection of shipping within spaceborne SAR imagery. The FSM(GP) algorithm has now its self 'evolved' into an algorithm known as EMMA[2]. EMMA is a simpler algorithm than the FSM(GP), in that the need for transition thresholds has been removed, and only one GP resides in each state. This simplification is highly desirable in military ATD algorithms where speed of execution is vital. The FSM(GP) performed classification via a mechanism that allowed GPs that were unsure of an objects classification to call other GPs for a 'second opinion'. This resulted in an implicit gathering of evidence. However, further use of the available information could have been made. For example, if a GP was assigning the object to class 1, but was unsure, causing a transition to another state, then what did it know about the object to have assigned it to class 1? A method of capturing this information was needed. EMMA achieves this through a powerful technique that explicitly passes classification information from state to state, and logically combines it to form a complex decision space. The time taken to train EMMA has also been drastically reduced via the introduction of a hybrid training scheme that comprises Dynamic Subset Sampling (DSS) [13, 14], and the Rational Allocation of Trials (RAT) [24].

A thorough description of EMMA's classification process is given in Section 5 on a synthetic data set. To demonstrate EMMA working with real world imagery, it is applied to the detection of microcalcifications in digitized mammograms. This also demonstrates that EMMA is a very versatile classification algorithm, and that its application is not restricted to military use.

[1] What is meant by 'reason' is clarified in Section 2.

[2] The reason for the name EMMA is discussed in Section 4.

2 Detection and Classification of Objects within Digital Images

Traditional decision theoretic methods strive to perform classification via the use of *discriminant functions*. Given M functions $d_1(\mathbf{x}), \ldots, d_M(\mathbf{x})$, M classes $\omega_1, \ldots, \omega_m$, and a vector of features $\mathbf{x} = (x_1, \ldots, x_n)$. The functions $d_1(\mathbf{x}), \ldots, d_M(\mathbf{x})$ are known as discriminant functions if $d_i(\mathbf{x}) > d_j(\mathbf{x})$, when $\mathbf{x}$ belongs to class ω_i. Common practice is to construct a decision boundary between two classes using the single discriminant function $d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$. This has the property that $d_{ij}(\mathbf{x}) > 0$ if $\mathbf{x}$ is from class ω_i, and $d_{ij}(\mathbf{x}) < 0$ if $\mathbf{x}$ is from class ω_j [15, pp 579–580]. This approach works well when two classes are linearly separable and unimodal. However, when two classes are not linearly separable, overlap, or are multimodal, deriving a discriminant function to discriminate between the two classes is neither trivial nor obvious. A classification system which is able to achieve this must be capable of identifying, and separating, each of the multiple clusters of each class.

Decision trees (DTs) [23] and Neural Networks (NN) [15, pp 595–619] are two commonly used methods that achieve this by constructing hyperplanes that partition the feature space. If the decision boundaries are highly complex, then decision surfaces composed of many intersecting hyperplanes must be formed. With NNs difficulty arises since hyperplanes do not simply stop at their intersection with other hyperplanes. As a result features of the same class may occur on both sides of the hyperplanes in feature space [15, p 617]. In addition, it is difficult to determine the correct NN architecture. DTs are capable of partitioning the feature space with

high accuracy, but the resulting DT can be very large with rule sets which are more complex than necessary [16]. Algorithms such as exhaustive search or dynamic programming can be used to construct DTs with minimum size and maximum accuracy, but are computationally infeasible except for trivial feature spaces [23]. There are many more classification algorithms, the number of which prevents discussion here. NNs and DT were mentioned since their method of splitting up the feature space most closely resembles the method of partitioning the feature space developed in this paper.

An algorithm that is capable of using a single discriminant function to partition a cluster of data from a class, rather than using many intersecting hyperplane would be very desirable. This function may then be interpreted and provide insight to the underlying distribution of the data, which can be difficult with DTs and impossible with NNs. Although desirable, what of multimodal data? Assume class ω_i is bimodal. Then more than one discriminant function may be required to classify each cluster. This is still simplistic in comparison to constructing multiple hyperplanes around each cluster. However, the algorithm would then require the ability to reason. In the bimodal case this could be as simple as: **if** the data sample is in the cluster defined by function one, **or** the data sample is in the cluster defined by function two, **then** the data sample belongs to class ω_i. For data with high modality, or which overlaps, as is the case for almost all real world data, constructing such a rule is not trivial. The algorithm would need to test a data sample against one of its discriminant functions, then make a decision as to whether to classify the sample based on what it knows so far, or whether more information is needed. If more information is needed, a decision must be made on how to combine this new information with what is already known. This implies that the algorithm must exhibit intelligent behaviour. An algorithm that is capable of this is developed in Section 4. But first some early works which share similarities with the algorithms developed in this paper are reviewed in Section 3.

3 Intelligence through Simulated Evolution

There are many lines of research in Artificial Intelligence (AI) such as Neural Networks (NN) and knowledge based systems to name but two. These systems strive to mimic human intellect in some form [11, p 2]. In the early sixties, Lawrence Fogel explored an alternative to using human intellect as a model for AI. Since human intelligence is a product of natural evolution, Fogel proposed creating artificial intelligence through simulated evolution [9, 10]. Fogel offered "intelligent behaviour is the composite ability to predict one's environment coupled with a translation of each prediction into a suitable response in light of some objective" [12, p 11]. Fogel et al. [12] performed a number of experiments in which the environment was modeled as a sequence of symbols from a finite alphabet. The aim of the work was to evolve an algorithm that could predict the next symbol to emerge from the environment, based on the sequence of symbols that came before it. Finite State Automata (FSA) were used by Fogel et al. as a useful representation of the algorithm. A small population[3] of FSA were exposed to the environment. The FSAs better able to predict the environment were retained and mutated to produce offspring that then replaced less able FSA. Fogel et al. termed this process Evolutionary Programming (EP).

Following these experiments other researchers such as Cornett [4], Cornett et al. [5], Trellue [25], and Atmar, [1], applied EP to pattern recognition problems. In these works FSA were evolved to perform character recognition of handwritten letters. The letters were digitized using various encoding schemes to provide a sequence of input symbols. For example Atmar [1, p 91] lays the letter on a 3×3 matrix, and quantizes each square of the matrix into one of the three symbols 1, 2, 3. The symbol 1 represents little or none of the letter present within the presently accessed square. The symbol 2 represents a moderate amount of the square occupied (30–60%), and the symbol 3 a great deal of the square occupied ($> 60\%$). Classification is determined by the final state of the FSA after being presented with the sequence.

The classification of handwritten letters in the aforementioned works was achieved since the FSA were able to adapt to, and predict their environment. As Fogel had proposed, AI could be realized through simulated evolution, rather than modeling human intelligence.

Humans however, poses a trait that is particular to them, and no other evolved life on earth. They do not simply adapt to meet the demands of the environment, they adapt and change the environment to meet their demands. Take for example a dam which stops the natural flow of a river to meet our demand for a power source. It may therefore be beneficial to evolve an algorithm that is able to adapt to the environment, and simultaneously change the environment to meet its own needs. This gaol is pursued in Section 4.

4 Environment Manipulating Mutable Automata (EMMA)

As with some of the works outlined in Section 3, the aim of this research is that of performing classification. More specifically, performing Automatic Target Detection within digital images. A target is an object within an image which we wish to detect. This may be vehicles in a rural scene, ships at sea, or cancerous growths in digitized mammograms. The environment of the algorithms in Section 3 was a sequence of input symbols. The environment of the ATD algorithm is the feature vector. It may be feasible to evolve ATD algorithms in the same vein as the algorithms of Section 3 by presenting each component of the vector in sequence, but this is not perused here. In this research the algorithms still take the form of mutable automata, but with the added ability of being

[3] Small population sizes were used due to the limited computational resources available at the time. A minimum population size of three machines was used [12, pp 27–38].

State	Input I	Output O_i	Next State q_n
q_k	F_k	$O_i = \begin{cases} O_{i-1} \text{ AND } F_k, \text{ if } F_k < 0 \\ O_{i-1} \text{ OR } \quad F_k, \text{ if } F_k \geq 0 \end{cases}$	$q_n = \begin{cases} q_m, \text{ if } F_k < 0 \\ q_h, \text{ if } F_k \geq 0 \end{cases}$

Table 1: Representation of a state q_k and its transitions. The logical functions AND, and OR were chosen arbitrarily for the purposes of demonstration.

able to manipulate both the sequence and number of symbols presented to it. The classification algorithm has thus been dubbed the Environment Manipulating Mutable Automata (EMMA). The environment manipulation is achieved by embedding a Genetic Program (GP) [17] in each of the automata's states. The GP uses the feature vector as its terminal set, and thus has the ability to select or discard appropriate or inappropriate features. This allows feature selection to be performed as the algorithm evolves, a process which normally needs to be performed explicitly before a classification algorithm is trained. The features are then combined by the GP function set to produce a discriminant function. The output of the GP is also used as the state input. Hence, the GP component of EMMA has the ability to manipulate the environment before it is presented to the automata component. EMMA may also halt and classify the target at any time it feels it has sufficient information on which to perform classification. Hence, EMMA may not require the full input sequence. In addition to a GP, each state has embedded within it, two logical functions. These logical functions are used to combine the discriminant function of the current state with the discriminant functions of previously visited states. This concatenation of the discriminant functions allows a highly complex decision space to be formed.

A description of how EMMA performs classification is now given. Assume the start state to be state 0 (denoted q_0). A feature vector $\mathbf{x} = (x_1, \ldots, x_n)$ to be classified is presented to EMMA. The state discriminant function F_0[4], which may use 1 to n of the features, is then evaluated and is treated as the state input, and, in the start state, as the state output. A positive return from F_0 is treated as a logical true (binary 1), and a negative return as a logical false (binary 0). A transition to another state, say q_k, is then executed based on the value returned by F_0. On entering q_k, F_k is executed, and again the value returned is mapped to 0 or 1. The returned value is then combined with the output of the last state via one of the state logical functions, and forms the state output. Again, a transition to a new state occurs based on the value returned by F_k. This procedure continues until the maximum number of user defined transitions has occurred, or the halt state is entered. The final output is used as the classification, 1 indicating target, and 0 indicating non-target. The functionality of a state and its associated transitions is depicted in Table 1.

The architecture of EMMA has a number of properties worthy of note. Classification of multimodal or overlapping data is obtainable. This is because no one discriminant function need be learnt for any one class. If for example a class is bimodal, a discriminant function for each of the clusters can be learnt and concatenated with a logical OR. To achieve this EMMA has the ability to make a decision as to which discriminant function to call next, and how to logically combine it with previously called discriminant functions, based on the classification currently being assigning to the object. In addition, since the discriminant functions are constructed from the features, two objects from the same class may cause a discriminant function to return very different values. This in turn influences EMMA's decision as to which discriminant function to call next, and consequently causes different logical combinations to be formed. Thus, no two objects from the same class need be classified using the same features. Instead the most appropriate features for that object are used. To the author's knowledge these properties, and EMMA's versatility, are not found explicitly in the literature, making this a novel approach. To clarify the classification process, a worked example is presented in Section 5.

[4]The GP embedded within its state.

5 Example Classification Problem

To enable visualization of the classification process, a two class example, with a two dimensional feature space is presented. The example chosen is that of classifying two intertwined spirals, which has been a challenge for pattern classification algorithms, and has been the subject of much work in the Neural Network community [3, 8, 18]. This problem has also been tackled using GP by Koza [17, pp 445–457]. Each spiral is composed of 97 points, has a radius of 6.5, and has 3 revolutions. For this problem the feature vector $\mathbf{x} = (x_1, x_2)$ consists of the x, y coordinates of the points that form the spirals. Figure 2(a) shows the spirals, in which the circles represent class one and the squares represent class two. In Figure 2 (e) and (f), an incorrectly classified point from class one is represented by a star, and an incorrectly classified point from class two is represented by a diamond. The GP terminal set used by EMMA for this problem was $\{$x, y, $\mathcal{R}$,$\}$, where x, y are the coordinates of a point on the spiral, and $\mathcal{R} \in (-1, 1)$ is a random constant. The GP, and state logical function sets used were $\{+, -, \times, \div, \sin, \cos\}$, and $\{$AND, OR, NAND, NOR, XOR$\}$. This is the same GP function and terminal set as used by Koza [17, p 446] with the omission of IFLTE (IF Less Than or Equal to). Koza included this function to provide GP with some decision making capability. Since this is an inherent strength of EMMA, the IFLTE function is not needed. The minimum and maximum number

$I : F_k \mapsto \begin{cases} 0 \text{ if } F_k < 0 \\ 1 \text{ if } F_k \geq 0 \end{cases}$	$c : O_i \mapsto O_{i-1}$ NAND I
$a : O_i \mapsto O_{i-1}$ AND I	$d : O_i \mapsto O_{i-1}$ NOR I
$b : O_i \mapsto O_{i-1}$ OR I	$e : O_i \mapsto O_{i-1}$ XOR I

Table 2: Mappings used for shorthand notation.

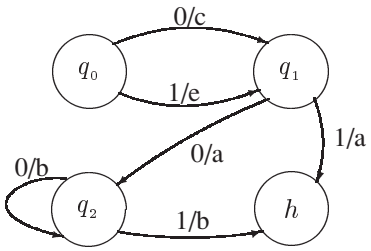
of states permissible was set to two and five respectively, and the maximum GP tree depth was set to four. Thirty runs of five hundred generations, with a population size of 252 individuals were performed.

Figure 1 shows an evolved four[5] state EMMA, and its functions F_0, F_1, and F_2, that classifies each point of the two spirals correctly. The mappings of Table 2 are used in Figure 1 to aid clarity. This EMMA is used to demonstrate step by step how classification of the two spirals is achieved. On entering the start state q_0, F_0 is executed and becomes both the input I and current output O_i. At this stage O_{i-1} and I cannot be logically combined by NAND or XOR (this states logical functions) since there has been no previous output O_{i-1}. A transition to state q_1 then occurs. On entering state q_1, F_1 is executed and provides the state input. If $F_1 \geq 0$ then the halt state is entered and the output is $O_i = O_{i-1}$ AND $F_1 \equiv F_0$ AND F_1. Referring to Figure 2 (c) and (e), the halt state is entered for all points which lay in the positive region defined by F_1, and their classification is given by F_0 AND F_1. All of these points are now correctly classified, and EMMA has determined that no further evaluation of them is necessary. If $F_1 < 0$ then a transition to state q_2 occurs, and the output is again F_0 AND F_1. On entering state q_2, F_2 is executed producing the state input. If $F_2 \geq 0$ then the halt state is entered and the output is $O_i = O_{i-1}$ OR $F_2 \equiv (F_0$ AND $F_1)$ OR F_2. Referring to Figure 2 (f), any point from class two that was incorrectly classified whilst in the previous state is now correctly classified. If $F_2 < 0$ then EMMA remains in the same state until the maximum number of allowed state transitions has been performed. EMMA then halts correctly classifying the points of class one which have not already been classified. A more efficient EMMA would have moved straight to the halt state for $F_2 < 0$ and still correctly classified the remaining points. However, no evolutionary pressure was applied to force halting before the maximum number of state transitions occurred, and so this repeated looping can be expected.

6 Training EMMA on Real World Imagery

A drawback of using Evolutionary Algorithms (EAs) is that they can be computationally expensive. The computational expense is proportional to $R \times G \times M \times N_s$, where R is the number of runs, G is the number of generations, M is the population size, and N_s is the number of training samples. EMMA performs classification on a pixel by pixel basis. Thus, the parameter N_s is a major contributor to the computational expense. If one 1024×1024 image is used for training purposes, then each population member must be evaluated $N_s = 1048576$ times each generation. This is infeasible if EMMA is to train within a reasonable time scale. To overcome this problem a combination of the Rational Allocation of Trials (RAT) [24], and an extension of Dynamic Subset Selection (DSS) [13, 14] is used to speed up the training of EMMA.

[5]There are in fact only three 'active' states, the fourth state is the halt state.



$$F_0 = 4.180636962 \sin\left(3.251366387\, x + \frac{y^2}{x}\right)$$

$$F_1 = \frac{(x - 0.205963\, y)\cos\left(\frac{\sin(0.192915\, x)}{\cos(x) - y + x}\right)}{(x + y)}$$

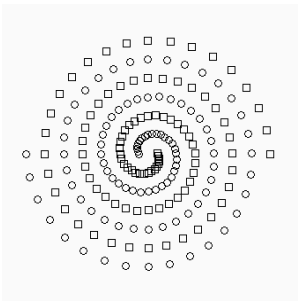
$$F_2 = \sin(3\, y - x)\,(-\cos(0.55221\, x) - \cos(\cos(xy)))$$

Figure 1: A four state EMMA that solves the spiral problem. The notation α/β denotes an input of α and an output of β, where the mappings of Table 2 are used for brevity. The start state is q_0.

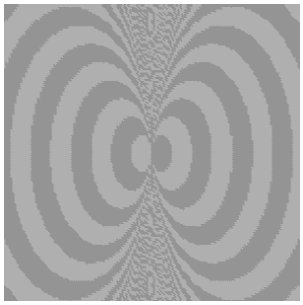
The RAT algorithm provides a method of speeding up the selection of one model amongst many. RAT is essentially the BRACE algorithm (RACEing with Blocking) [21] applied to EAs. The BRACE algorithm initially evaluates all N_m models on a small number of n_s training samples. It then 'races' the N_m models in parallel by evaluating each model on one training sample at a time. When it becomes statistically unlikely that a model will win the race, it is removed, and no further computation time is wasted evaluating it. Moreover, if it is clear that a model will win the race it too is removed. RAT applies the BRACE algorithm to EAs by assigning individuals to tournaments, and racing the tournament members.

The DSS algorithm randomly selects a subset of S_n samples from the whole training set each generation. The population is then evaluated on this subset only. The selection of a subset is weighted toward samples that the population has found difficult to classify, and samples that have not been seen for several generations. The DSS algorithm has been extended for use with EMMA so that two subsets are selected. A subset for class 1, and a subset for class 2, which ensures equal samples from both classes are selected.

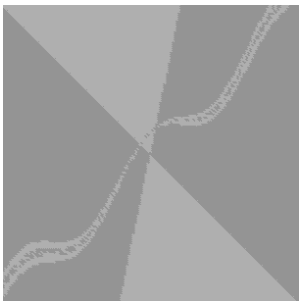
The training time of EMMA is reduced by combining the RAT and extended DSS algorithms. This combination shall



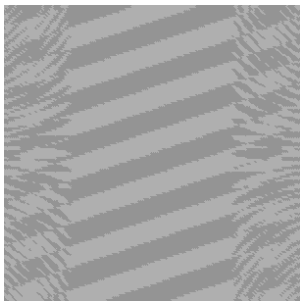
(a) Two spirals to classify



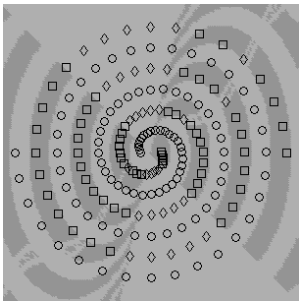
(b) F_0



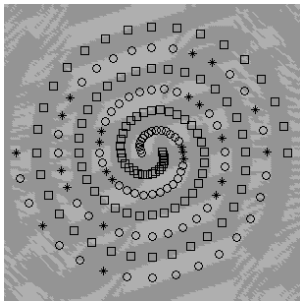
(c) F_1



(d) F_2



(e) F_0 AND F_1



(f) (F_0 AND F_1) OR F_2

Figure 2: The output of the EMMA shown in Figure 1. The dark regions represent a positive output from the function, and the light a negative.

be denoted DSSRAT, and is executed as follows. The population is split into μ tournaments. Two subset (one for each class) of size $\frac{S_n}{2}$ are selected from the training set, and then combined to form one subset. Each individual is then evaluated on the first n_s samples of the subset, after which RAT is used to determine if further evaluation of each individual is needed in order to produce a tournament winner. The maximum number of evaluations that any one individuals can receive is thus S_n.

7 Selection and Mutation

7.1 Selection

Selection is achieved via a $(\mu + \lambda)$-EP, with the slight difference that instead of conventional tournaments being used, DSSRAT is applied to the population to produce μ tournament winners. These μ individuals are retained as parents and produce $\lambda \geq \mu$ offspring. The union of the μ parents and λ offspring then form the next generation. In this research a population size of 252 individuals is used, with 25% of the population being retained as parents each generation. Thus, selection is via a (63+189)-EP, with four individuals competing in each tournament.

7.2 Mutation

There are a total of fourteen different mutations that may be applied to EMMA. These are: add a state, delete a state, change the start state, mutate a transition, cycle the states, headless chicken crossover on the states, exchange the GPs of two states, replace a state GP with a randomly created GP, mutate the state logical functions, headless chicken crossover on the GPs, grow a GP sub-tree, shrink a GP sub-tree, mutate a GP terminal, and mutate a GP function.

Choosing the frequency at which each mutation should be applied is very difficult due to their large number. To overcome this problem, the probability of each mutation occurring is evolved online. This is done at the individual level. Thus, each population member will have its own set of mutation probabilities, which are referred to as strategy parameters. The strategy parameters of a parent are mutated according to equation 1 before an offspring is produced. The new offspring is then created using the updated strategy parameters.

$$\begin{aligned} P_i' &= P_i \; e^{\tau' N(0,1) + \tau N_i(0,1)} \\ \tau &= \frac{1}{\sqrt{2\sqrt{n}}} \\ \tau' &= \frac{1}{\sqrt{2n}} \end{aligned} \tag{1}$$

where P' is the updated strategy parameters, n is the number of possible mutations, $N(0,1)$ is a normal random variable, and $N_i(0,1)$ is a normal random variable sampled anew for each of the possible mutations. It should be noted that equa-

		Ground Truth: Positive	Ground Truth: Negative
EMMA	Positive	TP true positive	FP false positive
	Negative	FN false negative	TN true negative

Table 3: Definitions used in pixel classification. If EMMA classifies a pixel as positive, and the pixel has been designated as positive in the ground truth, then this is a true positive.

tion 1 also provides a means of controlling the amount of mutations an individual receives. For example, if $P_i = 1 \ \forall \ i$ then the individual would receive n mutations (14 in our current case), and if $P_i = 0 \ \forall \ i$ it would receive none. If $P' > 1$ then it is reset to 1.

8 Detection of Microcalcifications in Digitized Mammograms

The detection of microcalcifications in digitized mammograms is used to demonstrate EMMA working with real world imagery. Microcalcifications are considered to be an important sign of breast cancer since they are found in 30%–40% of breast cancers detected radiographically in mammograms [19]. The size of microcalcifications are in the range of 0.1mm–1.0mm and have an average diameter of about 0.3mm, and appear on mammograms as small white spots. The mammograms used in this paper were obtained from the MIAS MiniMammographic Database[6] [7]. These images differ from the original MIAS database in that the original, digitized at 50 micron pixel edge, has been reduced to 200 micron pixel edge and clipped/padded so that every image is 1024×1024 pixels. Also provided with the images, are the center locations and radii of clusters of microcalcifications, rather than the locations of individual microcalcifications. Hence, before training EMMA, a ground truth of pixels containing microcalcifications was generated. These pixels are known as positives, and pixels containing no microcalcifications are known as negatives. For each pixel, a feature vector of sixteen features was calculated off line before training EMMA. At the beginning of each training run, the feature vectors were read from file and stored in memory. This procedure reduced training time since features are simply accessed from memory rather than recalculated each time they are needed. The features used were the pixel intensity, and the first three moments of a 3×3, 7×7, 9×9, and 15×15 window, centered on the pixel being classified. These feature vectors formed the GP terminal set, and the GP function set used was $\{+, -, \times, \div, \max, \min\}$, where max, and min are binary nodes returning the maximum and minimum of their arguments respectively. The state logical function set used was $\{$AND, OR, NAND, NOR, XOR$\}$. The minimum and maximum number of states was set to two and eight, and the maximum GP tree depth was set to five.

A problem faced when developing classification algorithms for use on real world imagery is that of obtaining a good trade off in sensitivity and specificity. Sensitivity, S_e, and specificity, S_p, are defined as:

$$S_e = \frac{TP}{TP + FN} \qquad S_p = \frac{TN}{TN + FP} \tag{2}$$

where TP, TN, FP, and FN are as defined in Table 3 [6, p 170]. It is desirable to maximize both the sensitivity and specificity. To this end the fitness function used was

$$f = S_e + S_p \qquad \text{where } f = \mathbb{R}_+ \in \{0, \ldots, 2\}. \tag{3}$$

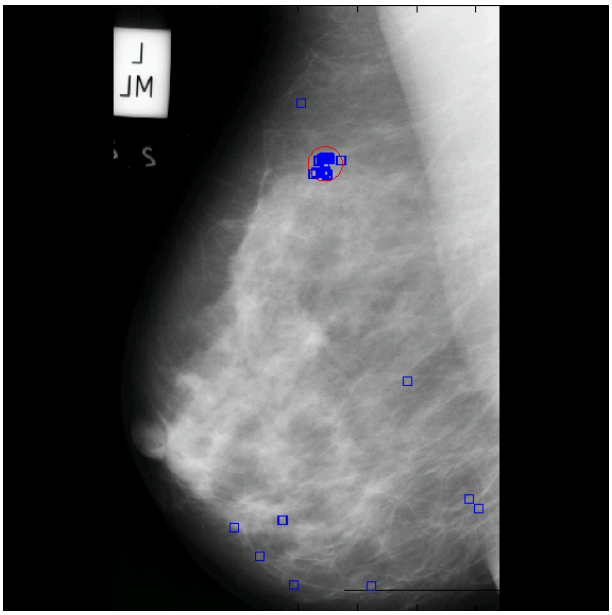
This fitness function behaved very well for the detection of microcalcifications. This is because early in the run EMMA found it relatively easy to maximize the sensitivity since the number of positives are very small in comparison with the negatives. Correctly classifying the positives is essential due to their link with breast cancers. As the run progressed, EMMA then began to work on maximizing the specificity, and hence minimize the false positives.

9 Experimental Results

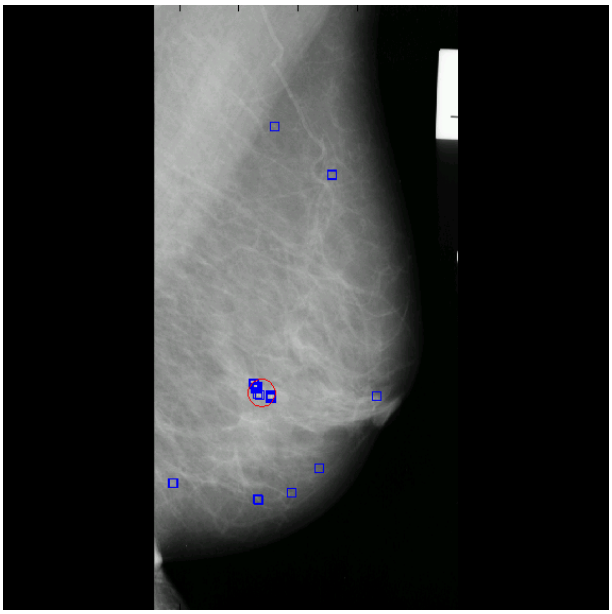
Thirty runs of two hundred generations were performed with EMMA training on one image. Each run took approximately forty minutes on a 450MHZ Pentium II PC. The best individual to emerge had six states and its output on two unseen test images is shown in Figure 3. EMMA has clearly identified **all** areas in which the radiologist has indicated there to be microcalcifications. This was the case for all seventeen unseen test images on which EMMA was run. Figure 3(a) also contains nine FPs, and Figure 3(b) contains six FPs. It should be noted however that these FPs are single pixels and not clusters. In other works such as [20], any single pixel classified as a microcalcification was removed, and in [26, p 883] only groups of three or more microcalcification were considered. If a similar procedure was applied to the results of EMMA, then the number of FPs would decrease dramatically, and in some cases, result in zero FPs. Since the number of FPs produced by EMMA is reasonably small, it may be prudent to leave them marked on the image, as in Figure 3, to let the radiologist decide whether these singletons are indeed FPs or microcalcification.

The detection of microcalcifications was used only as an illustration of EMMA's performance when applied to real world imagery. If EMMA was to be used by a radiologists as a diagnostic aid for the detection of microcalcifications, then further work would need to be carried out in order to obtain definitive results. For example, the full MIAS mammographic database containing 50μm $\times$ 50μm resolution images should be used. These images contain four times more

[6] Obtainable from http://peipa.essex.ac.uk/ipa/pix/mias/.



(a) Unseen test image.



(b) Unseen test image.

Figure 3: Typical output of EMMA on two unseen digitized mammograms. The squares are centered on the pixels EMMA has designated as microcalcification, and the larger circle in each image is the area in which the radiologist has indicated there to be microcalcifications.

information than the images used in this paper, and would allow fine grain features to be used. In addition, more than one training image should used as in [22], where 80% of the database is used for training and 20% for testing. This would provide EMMA with a more diverse and representative training set allowing better generalization. However, the results obtained in this very limited investigation are very promising indeed, and compare well with those reported in [20], [22] and [26]. Yoshida et al. [26, p 868] report a sensitivity of approximately 85% with a false positive rate of five clusters per image. Meesman et al. [20] show results indicating that 85%–90% of the clusters are detected with six FPs for selected regions of the mammogram. However, they quote "When we applied the networks to complete mammograms, the output of the networks contained a large amount of false positive clusters". Rosen et al. [22] report a better performance and were "able to find 91% of the clusters with a false positive rate of 0.13 clusters per image".

10 Summary

A novel approach to performing classification has been presented in an architecture dubbed EMMA. In this approach hypersurface discriminant functions are evolved using GP. These discriminant functions reside in the states of a FSA which is evolved simultaneously. The FSA component has the ability to reason, and combine the discriminant functions logically to produce complex decision spaces. No preprocessing need be performed before training EMMA, the raw images are simple presented. EMMA has the ability to perform feature selection for all its discriminant functions whilst it is evolving. Since each discriminant function will have different features, and one or more discriminant functions may be used to classify objects from the same class, no two objects from the same class need be classified using the same features. Instead, the most appropriate features for a given object are used to classify it. These properties make EMMA a very versatile ATD algorithm.

11 Acknowledgements

The Author would like to thank Colin Reeves, James Cubillo and David Booth for their support whilst carrying out this research. Thanks also go to David Fogel for answering many EC related questions over the last year, and for his efforts in supplying reference material used in this paper that would have otherwise been difficult to obtain.

Bibliography

[1] W.J Atmar. *Speculation on the Evolution of Intelligence and its Possible Realization in Machine Form.* Sc.D dissertation, New Mexico State University, Las Cruces, 1976.

[2] Karl Benson. Evolving finite state machines with embedded genetic programming for automatic target detection within SAR imagery. In *Proceedings of the Congress on Evolutionary Computation*, La Jolla Marriott, San Diego, USA, 16-19 July 2000. IEEE.

[3] G Carpenter, S Grossberg, N Markuzon, J Raynolds, and D Rosen. Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713, 1992.

[4] F N Cornett. An application of evolutionary programming to pattern recognition. MS thesis, New Mexico State University, Las Cruses, 1972.

[5] F N Cornett, V P Holmes, and D W Dearholt. Some experiments with evolutionary programs. In *Proceedings of 7th Annual Conference on Information Sciences and Systems*, March 1973.

[6] Russell C Eberhart and Roy W Dobbins. *NEURAL NETWORK PC TOOLS: A Practical Guide*. Academic Press, 1990.

[7] John Suckling et al. *The Mammographic Image Society Digital Mammogram Datatbase*, pages 375–378. International Congress Series 1069. Exerpta Medica, 1994.

[8] S E Fahlman and C Lebiere. The cascade-correlation learning architecture. In Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2. Morgan Kauffman, 1990.

[9] Lawrence J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

[10] Lawrence J. Fogel. *Biotechnology: Concepts and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1963.

[11] Lawrence J Fogel. *Intelligence Through Simulated Evolution: fourty years of evolutionary programming*. Wiley series on intelligent systems. Wiley, 1999.

[12] Lawrence J Fogel, Alvin J Owens, and Michel J Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[13] Chris Gathercole and Peter Ross. Some training subset selection methods for supervised learning in genetic programming. Presented at ECAI'94 Workshop on Applied Genetic and other Evolutionary Algorithms, 1994.

[14] Chris Gathercole and Peter Ross. Small populations over many generations can beat large populations over few generations in genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 111–118, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[15] Rafael C Gonzalez and Richard E Woods. *Digital Image Processing*. Addison Wessley, 1992.

[16] T K Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, August 1998.

[17] John R Koza. *Genetic Programming: on the programming of computers by means of natural selection*. The MIT Press, 1992.

[18] Kevin J Lang and Michael J Witbrock. Learning to tell two spirals apart. In *Proceedings of the 1988 Connectionist Summer Schools*. Morgan Kaufman, 1988.

[19] Huai Li, Ray K J Liu, and Shih-Chung B Lo. Fractal modeling and segmentation for the enhancement of microcalcifications in digital mammograms. Technical Research Report TR 96-38, Electrical Engineering Department and Institute for Systems Research, University of Maryland at College Park, 1996.

[20] D Meersman, P Scheunders, and D Van Dyck. Detection of microcalcifications using neural networks. In *3rd international workshop on digital mammography*, Chicago Illinois USA, 9–12 June 1996.

[21] Andrew W Moore and Mary S Lee. Efficient algorithms for minimizing cross validation error. In William W Cohen and Haym Hirish, editors, *Proceedings of the 11th International Conference on Machine Learning*, pages 190–198, Rutgers University, 1994. Morgan Kaufmann.

[22] Daniel Rosen, Benjamin Martin, Mark Monheit, Greg Wolff, and Martin Stanton. A baysian neural network to detect microcalcifications in digitized mammograms. In *3rd international workshop on digital mammography*, Chicago Illinois USA, 9–12 June 1996.

[23] S Rasoul Safavian and David Landgrebe. A survey of decicision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, May 1991.

[24] Astro Teller and David Andre. Automatically choosing the number of fitness cases: The rational allocation of trials. In *Proceedings of 2nd Annual Conference on Genetic Programming*, pages 321–328, Stanford University, CA, USA, 1997. Morgan Kaufmann.

[25] R E Trellue. The recognition of handprinted characters through evolutionary programming. MS thesis, New Mexico State University, Las Cruces, 1973.

[26] Hiroyuki Yoshida, Kunio Doi, and Robert M Nishikawa. Automated detection of clustered microcalcifications in digital mammograms using wavelet transform techniques. In *SPIE Image Processing*, volume 2167, 1994.