Multiple Species Weighted Voting – A Genetics-Based Machine Learning System

Alexander F. Tulai and Franz Oppacher

Computer Science Department, Carleton University,
Ottawa, Ontario, Canada, K1S 5B6
alex.tulai-rocaeng@rogers.com
oppacher@scs.carleton.ca

Abstract. Multiple Species Weighted Voting (MSWV) is a genetics-based machine learning (GBML) system with relatively few parameters that combines N two-class classifiers into an N -class classifier. MSWV uses two levels of *speciation*, one manual (a separate species is assigned to each two-class classifier) and one automatic, to reduce the size of the search space and also increase the accuracy of the decision rules discovered. The *population size* of each species is calculated based on the number of examples in the training set and each species is trained independently until a *stopping criterion* is met. During testing the algorithm uses a *weighted voting system* for predicting the class of an instance. MSWV can handle instances with unknown values and post pruning is not required. Using thirty-six real-world learning tasks we show that MSWV *significantly* outperforms a number of well known classification algorithms.

1 Introduction

Data classification, as a supervised learning task, has been one of the most researched subjects and the progress in this area has translated into a large variety of supervised learning algorithms. Among the genetics-base machine learning (GBML) systems, the more successful ones have been the learning classifier systems (LCS) with XCS being the main example [10, 11]. While Multiple Species Weighted Voting (MSWV) algorithm qualifies as a GBML because of its use of populations of individuals and two genetic operators, it is not an LCS.

The use of N 2-class classifiers in N -class classification tasks, that MSWV uses (discovered independently), is also the standard method used by Support Vector Machines [9]. There are also other algorithms using $N(N-1)/2$ SVM classifiers (one for each pair of classes) for an N -class classification task [5].

In section 2 we introduce a terminology and present a background of the classification theory while in section 3 we present the important techniques used by the algorithm. In section 4 we present the experimental results and compare them with well known competing techniques followed by conclusions in section 5.

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 1263–1274, 2004.

2 Background

A dataset D is a subset of $\mathcal{X} \times \mathcal{C}$ where $\mathcal{X}$ is the input space, also called the feature space, and $\mathcal{C}$ is the class space. In general $\mathcal{X} = X_1 \times \cdots \times X_n$ but in many practical applications $X_i = R$ so $\mathcal{X} = R^n$ and the feature space is n-dimensional. $\mathcal{C} = \{c_1, \ldots, c_N\}$, usually a discrete set of small cardinality N, is the class space. A record (instance or sample) of a data set, is a point $p = \{\mathbf{x}, c_i\} = \{x_1, \ldots, x_n, c_i\}$ where $\{x_j\}_{j=1,n}$ are the feature values and c_i is the class of the record (sometimes called the record tag).

Classification is the problem of predicting the value of an output variable $y \in \mathcal{C}$ based on a given value of the input variable $\mathbf{x} = \{x_1, \ldots, x_n\}$.

An attribute is a Boolean valued function $h_i : X_i \to \{0,1\}$. A decision rule r is a function $r : \{0,1\}^n \to \mathcal{C}$ usually built as a conjunction or disjunction of attributes. Decision rules are used in classification and they are usually constructed by a learning algorithm from a training set of L points $p_i = \{\mathbf{x}_i, y_i\}, i = 1, L$ of known input and output values. For a given point $\{\mathbf{x}, c_i\}$ with $\mathbf{x} = \{x_1, \ldots, x_n\}$ a correct classification occurs when $r(\{h_1(x_1), \ldots, h_n(x_n)\}) = \hat{y}(\mathrm{x}) = c_i$, otherwise ($\hat{y}(\mathbf{x}) \neq c_i$) it is considered an error. MSWV uses decision trees with internal nodes storing Boolean valued attributes and the leaf nodes containing a tag. Because the path from the root node to a leaf node is a conjunction of Boolean attributes, a decision tree can be seen as a disjunction of a limited number of conjunctions.

In general almost all learning algorithms are procedures for obtaining estimates $\{\hat{f}_i(x)\}_1^N$ of the set of conditional probabilities:

$$\{f_i(x) = \Pr(y = c_i \mid x)\}_1^N \tag{1.1}$$

There are two methods for obtaining these probability estimates [5], the density estimation approach and a regression methodology. The density estimation approach uses Bayes' theorem. A well known example of a classification algorithm using the density estimation paradigm is Naïve Bayes [6] but as referenced in [5] there are many others.

The second approach attempts to directly estimate the conditional probabilities (1.1). Examples of algorithms using the regression methodology are the decision tree induction methods [4, 8], the nearest neighbor methods [1] and others. MSWV also belongs to this class of algorithms. In the case of the algorithms using decision tree representations, f would be nothing else but a function built on the Boolean valued attributes $h(x)$ (like a decision rule). MSWV uses a GA to modify these decision trees and search for those trees that contain the best decision rules r giving us the best probability estimates $\{\hat{f}_i(\mathbf{x})\}_1^N$.

The predicted class $\hat{i}$ may be obtained from the probability estimates using:

$$\hat{i}(\mathbf{x}) = \arg \max_{1 \le i \le N} L_i \hat{f}_i(\mathbf{x}) \tag{1.2}$$

Very often $L_i = 1$ (see [5]) which is equivalent to assigning the predicted class to be the most probable for a given $\mathbf{x}$. In section 3.7, the equivalent of equation (1.2) will be given by equation (1.6) showing how the decision is made in the case of MSWV.

3 Algorithm Characteristics

In this section we detail the main characteristics of the MSWV system.

3.1 Representation and Species Definition

MSWV uses populations of individuals to create 2-class classifiers. The genotype and phenotype of an individual is a decision tree with internal nodes carrying Boolean valued attributes and leaf nodes carrying one of two possible class labels. What distinguishes MSWV from other algorithms using decision trees is the constraint we impose that each internal node must carry a Boolean valued attribute constructed from a *feature distinct* from the other internal nodes in the tree. This restriction implies that a decision tree can have a maximum of n internal nodes (where n, following the terminology in section 2, is the number of features) and a maximum of $n+1$ leaf nodes. Because the path from the root node to a leaf node represents a decision rule (as a conjunction of Boolean valued attributes) each decision tree can carry a maximum of $n+1$ decision rules. This restriction significantly reduces the size of the search space.

The decision trees are randomly initialized. Each internal node stores a Boolean valued attribute as a triplet $(\boldsymbol{feature\#}, \boldsymbol{featuretype}, \boldsymbol{values})$. The possible features and the attribute functions built on them are:

- *Boolean features (type 1).* $X_i = \{a, b\}$. An internal node stores $(i, 1, v)$ with $v \in \{a, b\}$, $h_i(x_i) = 1 \Leftrightarrow x_i = v$.
- *Set features (type 2).* $X_i = \{x_i^1, \cdots x_i^k\}$. An internal node stores $(i, 2, A)$ with $A \subseteq X_i$, $h_i(x_i^j) = 1 \Leftrightarrow x_i^j \in A$
- *Numerical features (Integer type 3, Floating point type 4).* $X_i = [x_i^{\min}, x_i^{\max}]$, $x_i \in Z \mid R$. A node stores $(i, 3 \mid 4, \{v_1, v_2\})$, $[v_1, v_2] \subseteq X_i$, $h_i(x_i) = 1 \Leftrightarrow x_i \in [v_1, v_2]$

Each 2-class classifier is represented by a population of decision trees with leaf nodes carrying one of two possible class labels, either a class $c_i \in C$ or the "don't know" class that we tag as "**?**" (i.e. $? = C - \{c_i\}$). A population of individuals recognizing the same two classes $\{c_i, ?\}$ forms a species, so each species is a 2-class classifier. For a N-class classification task, MSWV creates N species.

The training is done using the 1-v-r (one-versus-rest) method. For a given training set T the N classes induce a partition $T = T_1 \cup \ldots \cup T_N$ with T_i containing only the records tagged with c_i. The *i*-th species (corresponding to the *i*-th 2-class classifier)

is trained with the examples in T_i as positive examples and all other examples ($T - T_i$) as negative examples (treated as the "don't know" class).

The size of the *i*-th species population is automatically calculated with the formula:

$$m_i = \begin{cases} 20 & \textit{if } card(T_i)/2 < 20 \\ 200 & \textit{if } card(T_i)/2 > 200 \\ card(T_i)/2 & \textit{otherwise} \end{cases} \quad (1.3)$$

The reason behind using a population size of $card(T_i)/2$ is linked to the assumption that in the worst case the training subset T_i may contain just 2 examples for each possible classification rule that needs to be discovered in which case the maximum number of rules the *i*-th species should discover (although there is no guarantee they will be discovered) is $card(T_i)/2$. The upper limit of 200 on the population size is imposed by computational considerations. The lower limit of 20 is imposed by training time considerations and will be discussed in section 3.6.

3.2 Mutation Operators

In its current version MSWV uses only mutation and selection (described in the next section). Every generation all the individuals of a species (the species could be trained independently and in parallel) are mutated with probability 1, followed by selection. MSWV implements 6 mutation operators but with equal probability uses only one when mutation is applied. The six mutation operators are shown in Table 1. The term "randomly" refers to a uniform distribution random variable.

3.3 Rule Discovery and Automatic Speciation

One can classify the decision rules into *weak* and *strong* rules depending on whether they cover a small or a large number of training examples. There are datasets governed by one (or a very small number of) strong classification rule(s) in which case placing all the individuals in one large population and having them compete with each other by applying the selection operator seems to be the right way to go. Copies of the fittest individual will propagate through the population of decision trees resulting in an accelerated search for the strongest classification rule.

On the other hand, in general, the training examples are covered by a mixture of strong and weak rules [3]. This would suggest that following a simple hill climbing strategy by applying selection only between a parent and its offspring (we call this *restricted* selection) while slowing down the process of rule discovery, would ensure that both the strong and the weak rules were pursued by the evolving population.

MSWV achieves a compromise between the two extreme cases. Initially, each species starts with a population of hill climbers, individuals that mutate and use restricted selection to maintain the diversity of the evolved rules. Periodically the algorithm checks for *similar* individuals and places them together forming subspecies. The similarity is defined not in terms of genetic makeup but in terms of proficiency in

Table 1. The mutation operators used by MSWV.

Operator	Description
Add node	Adds a new internal node (and a new attribute)
Remove node	Randomly selects a leaf node and removes it and its parent node
Swap two leaves	Randomly selects a pair of sibling leaves and swaps them.
Swap internal nodes	Randomly selects two internal nodes and swaps them
Swap two branches	Randomly selects two internal nodes and swaps the sub-trees headed by them
Attribute change	*Boolean features*: the value used in the node is replaced by its complement
	Set features: With equal probability either adds one element to the set stored in the node or removes one
	Numerical features: Randomly either shift the range up | down or shrink | expand the range by a random gain

classifying positive and negative examples. The speciation process is formally described as follows. Let $\boldsymbol{S}_i$ represent the set of all individuals in species $\boldsymbol{i}$, trained with positive examples from the set $\boldsymbol{T}_i$ and negative examples from the set $\boldsymbol{T}-\boldsymbol{T}_i$. We introduce a pre-order relation on the elements of $\boldsymbol{S}_i$. If $s_1, s_2 \in \boldsymbol{S}_i$ are two individuals from species $\boldsymbol{i}$ and by $s_1(\boldsymbol{T}_i) \subseteq \boldsymbol{T}_i$, $s_1(\boldsymbol{T}-\boldsymbol{T}_i) \subseteq \boldsymbol{T}-\boldsymbol{T}_i$, $s_2(\boldsymbol{T}_i) \subseteq \boldsymbol{T}_i$ and $s_2(\boldsymbol{T}-\boldsymbol{T}_i) \subseteq \boldsymbol{T}-\boldsymbol{T}_i$ we denote the sets of positive and negative examples correctly classified by s_1 and s_2 we say that

$$s_1 \geq s_2 \Leftrightarrow s_1(\boldsymbol{T}_i) \supseteq s_2(\boldsymbol{T}_i) \;\&\; s_1(\boldsymbol{T}-\boldsymbol{T}_i) \supseteq s_2(\boldsymbol{T}-\boldsymbol{T}_i) \tag{1.4}$$

In other words, an individual is greater than or equal to another individual (from the same species) if and only if the first individual can correctly classify all the positive and all the negative training examples the second individual classifies. It can be easily shown that the relation "$\geq$" is reflexive and transitive but not antisymmetric. MSWV uses the pre-order relationship on the elements of $\boldsymbol{S}_i$ to create subspecies of individuals that are allowed to compete with each other and accelerate the search for specific decision rules. Because it is computationally intensive this operation (that we call *automatic speciation*) is not performed every generation but at regular intervals. All the experiments described in section 4 have used a speciation period of 50 generations. When the speciation operation is performed the second (or subsequent) time subspecies may already exist in the population. In this case only the fittest individuals from each subspecies participate in the speciation process in which case the pre-order relation between individuals will result in mergers between their respective subspecies. The use of the fittest individual of a subspecies is justified by the lack of diversity that characterizes the subspecies, as a result of full selection

pressure. We would like to point out that individuals of a subspecies may also be allowed to mate if an effective crossover operator is devised in the future.

3.4 Rule Confidence and Rule Weight

As previously mentioned, a decision tree encodes as many decision rules as there are leaf nodes in the tree. Although not mentioned in section 3.1, associated with each leaf there is some additional information built during the learning phase.

- A counter $\boldsymbol{P}$ that counts all the positive examples that have exercised the tree path (i.e. decision rule) ending at this leaf.
- A counter $\boldsymbol{N}$ that counts all the negative examples that have exercised the tree path ending at this leaf.
- A rule confidence coefficient calculated, following training, as follows:

  $\mu = \dfrac{\boldsymbol{P}}{\boldsymbol{P}+\boldsymbol{N}}$ (also called the true positive rate) for leaves tagged with $\boldsymbol{c}_i$

  $\mu = \dfrac{\boldsymbol{N}}{\boldsymbol{P}+\boldsymbol{N}}$ (also called the true negative rate) for leaves tagged with ?

The rule confidence coefficients are used to weight the individual vote during the testing phase (see section 3.7) and remove the need for post pruning.

Datasets often contain instances with unknown values, and learning algorithms must be able to deal with them with minimal degradation of performance.

MSWV treats the unknown (missing) values similar but not identical to other practical learning schemes like C4.5 [8] and PART [4]. When a record is evaluated (during training or testing) a path weight variable $\boldsymbol{pW}$, initially set to 1, is associated with the path taken by the record through the decision tree. Associated with each internal node of a decision tree there are 4 variables $(\boldsymbol{p}_l, \boldsymbol{p}_r, \boldsymbol{n}_l, \boldsymbol{n}_r)$ counting how many positive or negative examples have gone left or right. If an instance cannot be assigned deterministically to a branch because of an unknown value we assume that it may go on both branches and consequently two paths are created from that node down. The path weights of these two paths are updated using the 4 variables stored in the node with separate equations depending on whether we are testing or training.

During testing, the left and right path weight are given by $\boldsymbol{pW}\bullet\rho_l$ and $\boldsymbol{pW}\bullet\rho_r$ where

$\rho_l = (\boldsymbol{p}_l + \boldsymbol{n}_l)/(\boldsymbol{p}_l + \boldsymbol{p}_r + \boldsymbol{n}_l + \boldsymbol{n}_r)$, $\rho_r = (\boldsymbol{p}_r + \boldsymbol{n}_r)/(\boldsymbol{p}_l + \boldsymbol{p}_r + \boldsymbol{n}_l + \boldsymbol{n}_r)$

During training after calculating the path weight updating factors ρ_l and ρ_r we also have to (on-line) update the variables $(\boldsymbol{p}_l, \boldsymbol{p}_r)$ or $(\boldsymbol{n}_l, \boldsymbol{n}_r)$:

$\rho_l = \boldsymbol{p}_l/(\boldsymbol{p}_l + \boldsymbol{p}_r)$, $\rho_r = \boldsymbol{p}_r/(\boldsymbol{p}_l + \boldsymbol{p}_r)$, $\boldsymbol{p}_l = \boldsymbol{p}_l + \boldsymbol{pW}_l$, $\boldsymbol{p}_r = \boldsymbol{p}_r + \boldsymbol{pW}_r$

$\rho_l = \boldsymbol{n}_l/(\boldsymbol{n}_l + \boldsymbol{n}_r)$, $\rho_r = \boldsymbol{n}_r/(\boldsymbol{n}_l + \boldsymbol{n}_r)$, $\boldsymbol{n}_l = \boldsymbol{n}_l + \boldsymbol{pW}_l$, $\boldsymbol{n}_r = \boldsymbol{n}_r + \boldsymbol{pW}_r$

Because a path from the root node to a leaf node defines a rule, a path weight is nothing else but a rule weight. In section 3.5 we show how the path weights (or rule weights) are used during training (in fitness calculation) and in section 3.7 we show how they are used during testing (in vote weighting).

3.5 Individual Fitness Calculation

The fitness of an individual is calculated only once, when the individual is born, using the training data set. When during the process of evaluating a decision tree shaped chromosome on a training example a leaf node is reached the following updates take place. If the leaf has a tag c_i that matches the class of the example we increment the positive examples counter P and give a reward, otherwise we increment the negative examples counter N and give no reward. If the leaf is tagged "?" and the example is tagged c_i we increment the positive examples counter P and give no reward, otherwise we increment the negative examples counter N and give a reward.

The reward score for correctly classifying an example depends on the path weight. However we need to correct for the imbalance that may exist in the data set. If a class has very few representatives we need to give a higher reward for correctly classifying samples from this class. Consequently, the reward for correctly classifying a positive example $x \in T_i \subset T$ is $r_p(x) = \sum_{all\ paths} pW \bullet card(T) / card(T_i)$. The reward for classifying a negative example $x \in T - T_i$ (at a leaf node tagged with the unknown class "?") is given by the equation: $r_n(x) = \sum_{all\ paths} pW \bullet card(T) / card(T - T_i)$.

The sum over all paths is required for handling unknown values as explained in section 3.4. With these definitions, the fitness of an individual $s \in S_i$ is given by: $g_i(s) = (\sum_{x \in T_i} r_p(x) + \sum_{x \in T - T_i} r_n(x)) / 2$ (therefore $g_i(s) \in [0, card(T)]$).

One of the major concerns in machine learning is the generalization power of the rules learned. To encourage the emergence of short rules with a high generalization potential, MSWV reduces an individual's fitness by a penalty factor defined as: $\delta = 0.0005(n-1)L$ where n is the number of internal nodes in the tree (and distinct features used in the decision tree), L is the number of records in the training data set and the constant 0.0005 has been experimentally determined. Assuming a training dataset of 1000 instances, this small penalty factor implies that adding a new internal node to a decision tree must be justified by an increase in fitness greater than 0.5.

3.6 Training Time

Unlike other GBML systems that use a fixed, experimentally determined training time parameter [2, 3], MSWV uses a dynamic stopping criterion. The reasons why a stopping criterion is superior to a fixed training time method are:

- if the training phase stops too early the achieved rule accuracy may be too low
- a training phase that is too long may encourage data overfitting
- the "optimum" training time may greatly vary from dataset to dataset
- it improves the algorithm efficiency (stopped species do not consume CPU time)

Every Δ generations MSWV calculates the average fitness of all the individuals of a species. If the average fitness of all individuals of species i at time $t = j\Delta$ is given by $\bar{g}_i(j\Delta)$, the training will stop at time $j\Delta$ if the following condition is met

$$\frac{\overline{g}_i(j\Delta) - \overline{g}_i((j-3)\Delta)}{3} \leq 0.02 \max_{k \geq 1}(\overline{g}_i(k\Delta) - \overline{g}_i((k-1)\Delta)) \tag{1.5}$$

MSWV system uses a value of 10 for Δ. Put in words, a species is evolved until the increase in the (smoothed over 2Δ) average fitness of its individuals falls below 2% of the highest jump in average fitness over Δ generations. Because the highest jump in fitness usually occurs in the first Δ generations for almost all datasets it is reasonable to ensure the same behavior in the case of those datasets that have a very small number of examples for a certain class (these datasets are very rare but they do exist in the real-world). It is because of these special cases that we established a lower limit on population of 20 (see section 3.1) to increase the probability of positive evolutionary changes and a non zero average fitness increase early in the training.

3.7 Testing

Given an instance $\mathbf{x}$ from a testing dataset with N classes, MSWV uses a voting scheme to predict its class. Because the individuals in a subspecies are all very similar (due to the strong selection pressure) only the fittest individual in the subspecies is allowed to vote. Experiments performed with a very large number of datasets have shown there is no difference in results if all are allowed to vote or only the best is allowed to vote (but this method is more efficient). Let's assume species i finishes the training phase, with its population distributed in M_i subspecies. Let's also denote by M the maximum number of subspecies in any of the N species (i.e. $M = \max_{i=1,N} card(M_i)$). The outcome of the weighted voting process is given by the following formula (also see equation 1.2 for reference):

$$\hat{i}(\mathbf{x}) = \arg\max_{1 \leq i \leq N} \frac{M}{card(M_i)} \sum_{j=1}^{M_i} v_{ij}(\mathbf{x}) \tag{1.6}$$

where $v_{ij}(\mathbf{x})$ is the confidence and path (rule) weighted vote of the fittest individual from the j-th subspecies of the i-th species and is given by the formula:

$$v_{ij}(\mathbf{x}) = \sum_{all\ paths} (-1)^{1(leafTag == "?")} \mu \bullet pW, 1(x) = \begin{cases} 1 \, if \, x = true \\ 0 \, if \, x = false \end{cases} \tag{1.7}$$

where μ is the confidence coefficient calculated as explained in section 3.4, pW is the path weight and the summation over all possible paths is required for handling instances with unknowns (as explained in section 3.4). Confidence weighting removes the need for post pruning (for example, the vote of a rule with confidence 0, is 0).

4 Experimental Results

The performance of MSWV has been evaluated on a set of thirty-six standard datasets available from UCI at: www.ics.uci.edu/~mlearn/MLRepository.html. The thirty-six datasets used for the study exhibit a large variety of characteristics in terms of number of instances, number and diversity of features, number of classes, missing feature values, etc. The datasets and their characteristics are listed in Table 2.

Table 2. The datasets used for experiments (unknown values given as a percentage of total).

Dataset	Id	Inst	Unknown	#feat	#nom	#num	#class
Audiology	aud	226	2.0	69	69		24
Balance	bal	625	-	4		4	3
Breast-w	brw	699	0.3	9		9	2
Bupa	bpa	345	-	6		6	2
Contraceptive	cmc	1473	-	9	4	5	3
Credit-austral.	cra	690	0.6	15	9	6	2
Credit-german	crg	1000	-	20	13	7	2
Glass	gls	214	-	9		9	6
Heart-c	h-c	303	0.2	13	7	6	2
Heart-h	h-h	294	20.5	13	7	6	2
Hepatitis	hep	155	5.6	19	12	7	2
Ionosphere	ion	351	-	34		34	2
Iris	irs	150	-	4		4	3
Labor	lbr	57	35.7	16	8	8	2
Pima-indians	pmi	768	-	8		8	2
Primary-tumor	prt	339	3.9	17	17		22
Sonar	snr	208	-	60		60	2
Soybean	soy	683	9.8	35	35		19
Vehicle	veh	846	-	18		18	4
Vote	vot	435	5.6	16	16		2
Vowel	vow	990	-	13	3	10	10
Wine	wne	178	-	13		13	3
Yeast	yst	1484	-	8		8	10
Zoo	zoo	101	-	16	16		7
Adult	ADL	48842	0.9	14	8	6	2
Hypothyroid	HTH	3772	5.5	29	22	7	4
Kr-vs-kp	KRK	3196	-	36	36		2
Led (10%noise)	LED	6000	-	7	7		10
Letter	LTT	20000	-	16		16	26
Mushrooms	MUS	8124	1.4	22	22		2
Optical-digits	ODG	5620	-	64		64	10
Satellite-image	SAT	6435	-	36		36	6
Segment	SEG	2310	-	19		19	7
Sick	SIC	3772	5.5	29	22	7	2
Splice	SPL	3190	-	60	60		3
Waveform+noise	WVF	5000	-	40		40	3

The performance of MSWV is compared to six other learning algorithms: NB [6], IB1 and IB3 (IBk or Instance-Based learner [1] that we use with k=1,3) , C4.5 (revision 8, an induction tree algorithm, [8]) , PART (an algorithm for inferring rules by repeatedly generating partial decision trees, [4]) and SMO (Sequential Minimal Optimization, a SVM classifier system [7]), all of them run using the Weka v3.4 package [12] available at: www.cs.waikato.ac.nz/ml/weka

The datasets are divided in two groups, twenty-four small (under 2000 instances) datasets and twelve large (over 2000 instances) datasets.

On the group of small datasets (lower case IDs) the comparison between classifiers is based on averaging ten ten-fold cross-validation runs. To test the statistical significance of the differences between classifiers on this group of datasets we used a paired two-sided *t*-test at 99% confidence level.

On the group of large datasets (upper case IDs) the comparison is based on the holdout estimate, where 33.3% of the instances are used for training and 66.6% are used for testing. The statistical significance of the differences between classifiers is performed using a test for the differences of two proportions at 99% confidence level. The large datasets have been shuffled offline and we have ensured that all classification systems have used the same partitions for training and testing.

Finally, the paired Wilcoxon signed rank test is used to calculate the statistical significance of the overall observed differences between two learning systems.

The accuracy rates (given as percentage of correct classifications) of the classification systems are summarized in Table 3. Results for the six learning systems used in the study are marked with ○ if they show significant improvement over the corresponding results for MSWV, and with ● if they show significant degradation.

The results presented in Table 3 can serve as basis for several observations. The first observation is that on this set of thirty-six datasets MSWV significantly outperforms 4 other classification systems (NB, IB1, C4.5 and PART), slightly outperforms IB3 and it is outperformed by SMO. This conclusion is based on the results presented in the row labeled W-L-T (wins-losses-ties) and also from the outcome of the paired Wilcoxon signed rank test as shown in the row labeled "Confidence". For example, MSWV significantly outperforms C4.5 on 17 datasets and it is significantly outperformed by C4.5 on 7 datasets. Using the results of the Wilcoxon test we can say that on this group of datasets MSWV improves C4.5 with 97.6% confidence but it is also improved by SMO with 54.7% confidence. The comparison with the other classification systems is done similarly.

Using a similar methodology and the experimental results on thirty datasets, [2] reported that XCS and their own UCS (both are Learning Classifier Systems) were outperformed by C4.5, PART and IB3. The SMO classifier used in our study could handle nominal and numerical valued features (different from the version used in [2]). We can indirectly conclude that MSWV may outperform XCS and UCS on a subgroup of datasets. In addition to that, MSWV does not exhibit the deterioration in performance reported in [2] on datasets with high number of classes like *aud*, *soy* or *vow* (see Table 2 for IDs and Table 3 for MSWV's results).

As shown in Table 3, MSWV is outperformed by all other six classifier systems on one single dataset, *ADL* that happens to be the dataset with the highest number of instances. Future experiments on datasets with very large number of instances (> 20,000) will help clarifying whether the performance degrades for very large datasets.

MSWV has outperformed all other classifiers on two datasets, *gls* and *yst* that have continuous numerical features and more than two classes. This may suggest that either MSWV does particularly well on this kind of datasets or, alternatively, the performance of the other six classifiers on datasets with continuous numerical features does not match their performance on datasets with integer or nominal features (this second alternative may be particularly true for the SMO classifier).

Table 3. Accuracy rates (%) of the classifiers used in the study. The row labelled W-L-T counts the wins-losses-ties of one-on-one comparison between MSWV and another classifier. The last two rows show the z-value and the confidence level of the Wilcoxon signed rank test.

	MSWV	NB		IB1		IB3		C4.5		PART		SMO	
aud	76.86	72.61	●	75.00		78.41		77.26		79.42	○	80.75	○
bal	86.62	90.53	○	79.28	●	86.74		77.82	●	83.17	●	87.42	○
brw	96.91	96.07	●	95.44	●	96.61		95.01	●	94.69	●	96.75	
bpa	67.39	54.87	●	62.20	●	62.49	●	65.83		65.25	●	58.00	●
cmc	52.99	49.72	●	42.49	●	45.11	●	52.73		50.17	●	49.82	●
cra	85.72	77.86	●	81.57	●	84.96	●	85.57		84.45	●	84.88	●
crg	73.59	75.16	○	71.88	●	72.21	●	71.25	●	70.54	●	75.15	○
gls	72.01	49.44	●	69.95	●	70.00	●	67.62	●	68.74	●	57.34	●
h-c	82.44	83.33	○	76.04	●	81.82		76.93	●	77.99	●	83.89	○
h-h	83.40	83.95		78.33	●	82.07	●	80.20	●	81.12	●	82.79	
hep	84.26	83.81		81.35	●	80.84	●	79.23	●	79.74	●	85.68	○
ion	91.03	82.17	●	87.09	●	86.01	●	89.74	●	90.83		88.06	●
irs	94.80	95.53		95.40		95.20		94.73		94.20		96.20	○
lbr	84.39	93.51	○	84.21		92.81	○	78.77	●	77.89	●	92.98	○
pmi	74.61	75.76	○	70.62	●	73.87		74.49		73.46		76.80	○
prt	44.96	49.71	○	34.34	●	44.99		41.39	●	40.86	●	46.96	○
snr	80.53	67.69	●	86.15	○	83.75	○	73.61	●	77.40	●	76.49	●
soy	91.40	92.81	○	90.07	●	91.07		91.65		91.27		92.97	○
veh	67.55	44.68	●	69.59	○	70.21	○	72.28	○	72.21	○	74.08	○
vow	87.47	62.90	●	99.05	○	96.99	○	80.20	●	77.67	●	70.59	●
vot	95.08	90.02	●	92.44	●	93.08	●	96.57	○	95.98	○	95.77	○
wne	96.01	97.47	○	95.11		95.84		93.20	●	92.08	●	98.76	○
yst	58.76	57.99	●	52.61	●	55.09	●	56.61	●	54.81	●	57.10	●
zoo	93.86	94.95		96.53	○	95.54		92.57		93.37		96.04	○
ADL	75.91	83.38	○	78.75	○	81.74	○	85.92	○	84.80	○	84.98	○
HTH	93.52	95.63	○	90.66	●	92.64		99.20	○	99.20	○	93.83	
KRK	94.60	84.61	●	87.56	●	92.07	●	99.01	○	98.22	○	94.23	
LED	74.33	74.08		62.48	●	74.25		74.28		74.23		74.03	
LTT	76.29	64.20	●	93.02	○	91.55	○	81.33	○	82.42	○	79.18	○
MUS	99.89	94.43	●	100.0		99.91		100.0		99.88		100.0	
ODG	92.93	90.71	●	97.84	○	98.05	○	86.55	●	87.70	●	97.12	○
SAT	61.91	61.42		69.04	○	69.53	○	59.62		61.35		69.63	○
SEG	92.98	79.22	●	95.52	○	94.35		93.50		93.90		90.71	
SIC	97.10	93.04	●	95.30	●	95.94		98.25	○	97.89		93.76	●
SPL	95.63	94.54		73.01	●	77.39	●	92.05	●	90.79	●	93.04	●
WVF	82.27	79.75		71.29	●	75.88	●	73.45	●	76.63	●	85.99	○
W-L-T		18-10-8		22-9-5		13-8-15		17-7-12		19-7-10		10-19-7	
z-value		2.31		2.2		0.31		2.26		2.26		-0.75	
Confid.		97.91		97.22		24.34		97.62		97.62		-54.67	

The performance of MSWV on two large datasets that have noise artificially added, *LED* and *WVF*, is also good, as shown in Table 3.

5 Conclusions

MSWV uses a GA to evolve populations of decision trees (each population a 2-class classifier) but uses a calculated population size and a training stopping criterion rather than some fixed values. Other variables (like speciation period or fitness averaging interval) have been kept constant in all our experiments supporting the claim that MSWV is a relatively parameter free GBML system. An innovative speciation mechanism ensures that strong and weak rules are pursued by the evolving population at the best pace possible. The restriction that the decision tree be constructed on attributes built on distinct features clearly helps the search but seems to introduce a limit on the complexity of the decision rules that can emerge in such trees and to negatively impact the classifier accuracy. This potential problem is successfully compensated by the statistical power of a voting system, making classification a collective task. Tested on a large number of real world classification tasks the algorithm significantly outperforms a number of well known classification algorithms. MSWV performs well on noisy datasets, can handle records with unknown values and does not require post pruning. Based on the experimental results we believe that at this moment MSWV is the best GBML system reported in the EC literature.

References

1. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning vol. 6, (1991) 37-66
2. Bernadó-Mansilla, E., Garrell-Guiu, J.M.: Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evol. Comput.11(3) (2003) 209-238
3. Carvalho, D.R. , Freitas A.A.: A genetic algorithm with sequential niching for discovering small-disjunct rules. Genetic and Evolutionary Computation Conference proceedings. Morgan Kauffman publishers, San Francisco, CA (2002).
4. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: Shavlik, J., editor, Machine Learning proceedings of the 15-th international conference. Morgan Kaufmann (1998) 144-151
5. Friedman, J.H., Another approach to polychotomous classification. Technical report, Stanford department of Statistics [www-stat.stanford.edu/reports/friedman/poly.ps.Z] (1996)
6. John, G., Langley, P.: Estimating Continuous Distributions in Bayesian Classifiers**.** In: P. Besnard and S. Hanks (Eds.) 11-th Annual Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo (1995)
7. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods – Support Vector Learning. Cambridge, MA:MIT Press (1998)
8. Quinlan, J.R. C4.5: Programs for Machine Learning, Morgan Kaufmann Publisher (1993)
9. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York, NY, USA (1998)
10. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation (1995), 3(2) 149-175
11. Wilson, S.W.: Generalization in the XCS classifier system. In: Genetic Programming Proceedings of the Third Annual Conference, J. Koza et al., eds., San Francisco, CA: Morgan Kaufmann (1998) 665-674
12. Witten, I.H., Frank, E.: Data mining practical Machine Learning tools and techniques with Java implementations. Morgan Kauffman Publishers (2000)