Lecture 10: Genetic Programming

CS408: Evolutionary Computation and Its Applications

Xin Yao

CSE, SUSTech

8 May 2023

Review of the Last Lecture

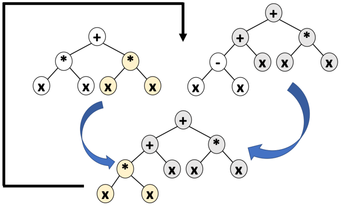
- **Penalty Methods**
- **Stochastic Ranking**
- **Repair Functions**
- **Specialised Representations**
- **Decoder Functions**

Outline of This Lecture

Genetic Programming (GP)

First used by de Garis to indicate the evolution of artificial neural networks [3], but used by Koza to indicate the application of GP to the evolution of computer programs [6].

1. **Trees (especially Lisp expression trees) are often used to represent individuals.**
2. **Both crossover and mutation are used.**

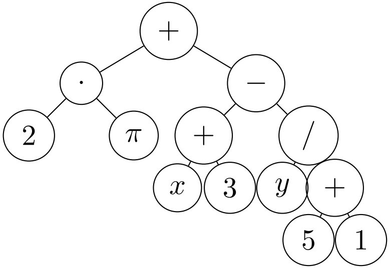


Representation	Tree structures
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

Figure 1: Left: illustration of GP. Right: Table 6.4 of [4].

Tree Representation: An Example

- **Arithmetic formula:** $2 \cdot \pi + ((x+3) - \frac{y}{5+1})$
- **Prefix notation**
  - **Polish notation:** $+(\cdot(2,\pi), -(+(x,3), /(y, +(5,1))))$
  - **LISP notation:** $(+(\cdot\ 2\ \pi)\ (-(+\ x\ 3)(/\ y\ (+\ 5\ 1))))$
- **Parse tree:**
  - **Nodes (or points) indicate the instructions to execute.**
    - **Internal nodes: functions.**
    - **Leaves: terminals.**
  - **Links indicate the arguments for each instrucption.**



Outline of This Lecture

Introduction

Main Steps of GP and Its Operators

Evolving Boolean N-multiplexer Functions using GP

Multi-objective Genetic Programming (MOGP) Approaches

Evaluation of MOGP Fitness Schemes

Automatic Ensemble Selection using MOGP

MOGP for Maximising ROC Performance

Preparatory Steps [7]

Before all, let's see what you need to prepare first . . .

1. **The set of primitive functions F. $\rightarrow$ Define the search space.**
   - **Example: arithmetic functions and conditional branching operators, or actions of a robot.**
2. **The set of terminals T.$\rightarrow$ Define the search space.**
   - **Example: independent variables (program's external input), zero-argument functions and numerical constants.**
3. **The (explicit or implicit) fitness function.$\rightarrow$ Define the goal.**
4. **The algorithm control parameters.**
   - **Example: population size and maximum size for programs.**
5. **The termination criterion/criteria.**
   - **Example: maximum number of generations, or problem-specific success predicate.**

Algorithm 1 Main Steps of GP [7].

```
Input: a set of functions F
Input: a set of terminals T
Input: a fitness measure
Input: control parameters
Randomly initialise a population of individuals pop
t = 0
while termination criterion is not met do
    for program ∈ pop do                                          ▶ Evaluation
        Execute individual program and obtain its fitness
    end for
    Select one or two individual(s) from pop with a probability based on fitness (with re-selection allowed)
    Create new individual(s) for the population by applying reproduction, crossover, mutation and architecture-altering
operations with specified probabilities
end while
Return the best-so-far individual
```

Now, let's go through *initialisation, crossover, mutation and architecture-altering operators* one by one.

Applications of GP [10]

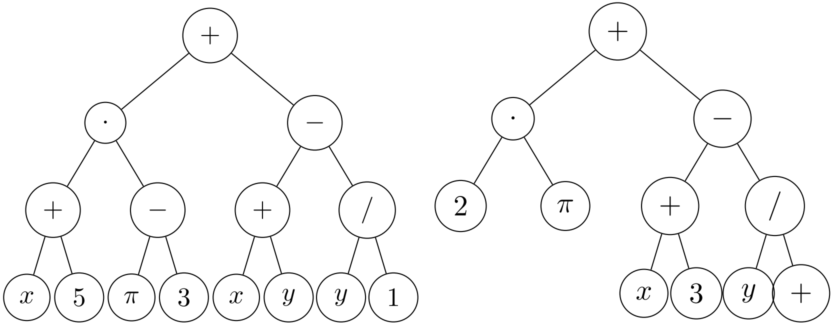
- **Systems Modelling (e.g., approximation of models).**
- **Design (e.g., evolve structure of monomer/circuit).**
- **Control (e.g., robot control).**
- **Optimisation and scheduling (e.g., GP based facility layout scheme).**
- **Signal processing (e.g., evolve adaptive digital signal processing algorithms).**

$\rightarrow$ **Seek models with maximum fit; evolve the syntax of arithmetic expressions, formulas in first-order predicate logic, or programs.**

John R Koza, Forrest H Bennett III, and David Andre. *Method and apparatus for automated design of complex structures using genetic programming*. US Patent 5,867,397. 1999

Initialisation Methods I

- **Full method (left): all the branches has depth D_{max}.**
- **Grow method (right): the branches may have different depths.**



Initialisation Methods II

- **Ramped half-and-half (混合法):**
  - **Half of the population are generated using the full method, while using grow method for generating the others.**
  - **Sometimes, each individual is generated using either method with equal probability.**

Algorithm 2 Ramped half-and-half.

Input: a set of functions F
Input: a set of terminals T
Input: maximum depth D_{max}
for $i \in \{1, \dots, \mu\}$ **do**
    if rand < 0.5 **then** ▶ Full method
        for $d \in \{1, \dots, D_{max} - 1\}$ **do**
            The contents of nodes at depth d are chosen from F ▶ Select functions
        end for
        The contents of nodes at depth D_{max} are chosen from T ▶ Select terminals at leaves
    else ▶ Grow method
        The tree is constructed beginning from the root, with the contents of a node being chosen stochastically from $F \cup T$ if $d < D_{max}$
    end if
end for

Crossover & Mutation

Crossover **Create new offspring by recombining randomly chosen parts from two selected parents.**

Mutation **Create one new offspring by randomly mutating a randomly chosen part of one selected parent.**

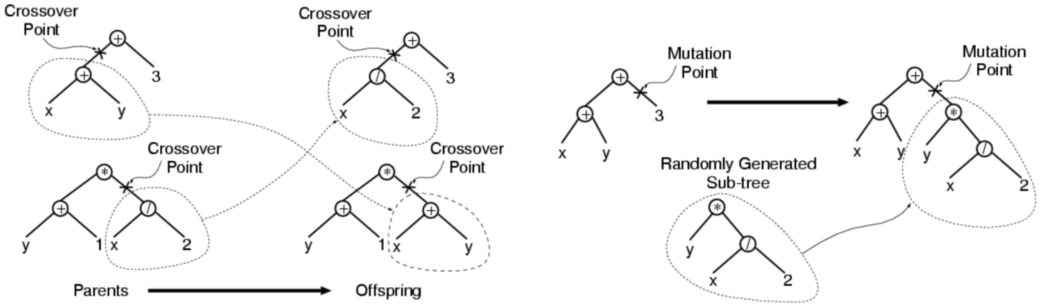


Figure 2: Figures 5.10 and 5.11 of [7].

Selection I

- **Over-selection**
  - **Often used to deal with the typically large population sizes.**
  - **Large populations are not unusual in GP.**
  - **Steps:**
    1. **Rank the population.**
    2. **Divide the population into two groups: G_A contains the top $\alpha\%$ and G_B contains the $(100-\alpha)\%$.**
    3. **Parent selection: 80% come from G_A while the rest come from G_B.**
  - **How to select α?**
    - **α is found empirically.**
    - **α depends on the population size.**
    - **The selection pressure increases dramatically for larger populations.**
      $\rightarrow$ #individuals from which the majority of parents are chosen stays a low constant value.

Selection II

- **Bloat (膨胀): Average tree sizes tend to grow along the evolution.**
  - **Also called the *code growth* or *Survival of the fattest*.**
  - **Questions: Why does bloat happen? Is bloat good?**
  - **Main techniques to control bloat:**
    1. **Parsimony pressure: Penalty term to reduce the fitness of large trees.**
    2. **Set a fixed limit on the size or depth of the tree.**
       $\rightarrow$ Reject a tree if it is over-sized (consider as an infeasible solution). $\rightarrow$ Or set fitness to 0 if over-sized.
    3. **Modify search operators.**
    4. **Multi-objective techniques (fitness and size).**

  All sounds like Constraint Handling techniques!

Outline of This Lecture

Evolving Boolean N-multiplexer Functions

- **Input:** k **address bits** a_i **and** 2^k **data bits** d_i**, where** $N = k + 2^k$**.**

$$a_{k-1}, \ldots, a_1, a_0, d_{2^k-1}, \ldots, d_1, d_0.$$

- **Output of Boolean multiplexer is the particular data bit that is singled out by the address bits (e.g., see Figure).**

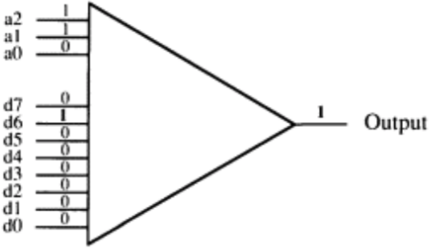


Figure 3: Figure 7.27 of [7]: Boolean 11-multiplexer with input of 11001000000 and output of 1, as the address bits are 110, which refers to d_6.

Steps for Evolving a Boolean 11-multiplexer

1. **Select the set of functions:**

$$F = \{\texttt{AND}, \texttt{OR}, \texttt{NOT}, \texttt{IF}\}$$

2. **Select the set of terminals: 3+8=11 arguments.**

$$T = \{a_0, a_1, a_2, d_0, d_1, \ldots, d_7\}.$$

3. **Identify the fitness measure for any evolved multiplexer m:**
   - $fitness_{raw}(m) =$ **the number of $\mathbf{x}$ that $m(\mathbf{x}) == TrueValue(\mathbf{x})$.**

4. **Select the values of control parameters: $PopSize = 4000$.**

5. **Specify the criterion for designating a result and the termination criterion.**

Summary Tableau for the Boolean 11-multiplexer

Objective:	Find a Boolean S-expression whose output is the same as the Boolean 11-multiplexer function.
Terminal set:	`A0, A1, A2, D0, D1, D2, D3, D4, D5, D6, D7.`
Function set:	`AND, OR, NOT, IF.`
Fitness cases:	The $2^{11} = 2{,}048$ combinations of the 11 Boolean arguments.
Raw fitness:	Number of fitness cases for which the S-expression matches correct output.
Standardized fitness:	Sum, taken over the $2^{11} = 2{,}048$ fitness cases, of the Hamming distances (i.e., number of mismatches). Standardized fitness equals 2,048 minus raw fitness for this problem.
Hits:	Equivalent to raw fitness for this problem.
Wrapper:	None.
Parameters:	$M = 4{,}000$ (with over-selection). $G = 51$.
Success predicate:	An S-expression scores 2,048 hits.

Figure 4: Table 7.6 of [7]: Tableau for the Boolean 11-multiplexer problem.

Hits Histograms of Fitness Values

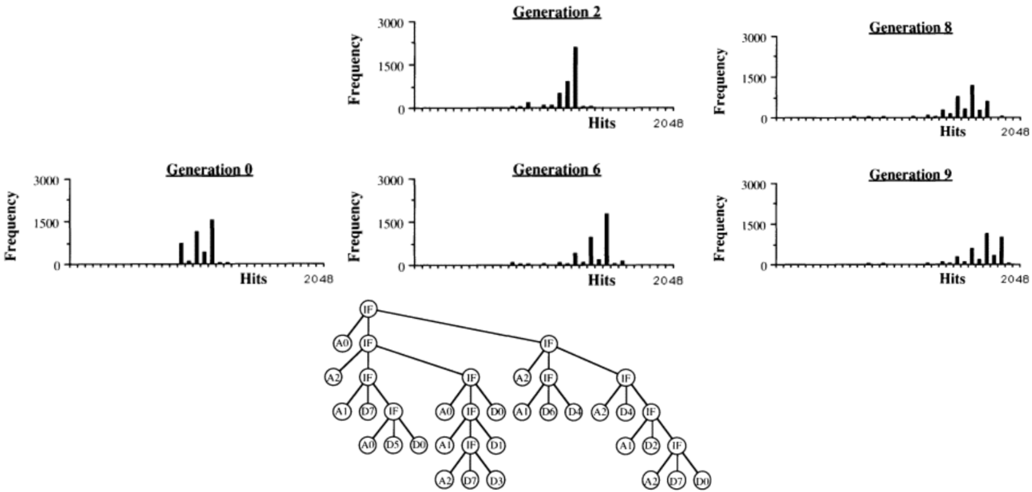


Figure 5: Figures 7.34 and 7.33 of [7]: Hits histograms for generations 0, 2, 6, 8, and 9 for the 11-multiplexer problem, and the best-of-run individual from generation 9 which solves the problem.

Exercise

You will work on evolving functions for N-Parity problems in today's lab.

Outline of This Lecture

Motivation: Class Imbalance I

- Given a classification task, most machine learning (ML) methods assume that:
  - Every class has the same misclassification cost.
  - The distribution of data among different classes is balanced.
  - The aim is to maximise the classification accuracy: $Acc = \frac{TP+TN}{TP+TN+FP+FN}$.

		Actual: Positive	Actual: Negative
Prediction	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

- Question: Is the following prediction good ($Acc = 98.92\%$)?

	Positive	Negative
Positive	10,000	10
Negative	99	1

Motivation: Class Imbalance II

- **Many real-world applications have very unbalanced distributions among classes, such as medical diagnostics, fraud detection, fault detection. Sometimes, $TNR = \frac{TN}{FN+TN}$ is more important than $TPR = \frac{TP}{FP+TP}$.**
- **Minority class: rare cases, high misclassification cost.**
- **Quantifying cost is hard in practice.**

Core Ingredient: GP to Evolve Classifiers

- **A classifier is a mapping from an instance to a class ID.**
  $\rightarrow$ **A mapping can be a function and `if-else` blocks.**
  $\rightarrow$ **A function and `if-else` blocks can be represented as a tree.**
  $\rightarrow$ **A tree can be evolved by GP.**

  $\rightarrow$ **GP can be used to evolve classifiers.**

- **However, GP can also evolve classifiers biased toward the majority class when data are unbalanced.**

Let's Put All Together

- **GP**
- **Ensemble learning**
- **Multi-objective optimisation**

$\Longrightarrow$ **Urvesh Bhowan et al. "Evolving diverse ensembles using genetic programming for classification with unbalanced data". In: *IEEE Transactions on Evolutionary Computation* 17.3 (2013), pp. 368–386**

Objectives and Fitness Design

- **Baseline: Single-objective GP (SOGP).**
  - **Fitness:** $Ave = \omega TPR + (1 - \omega) TNR$**.**
    - **Assume positive samples are majority.**
    - **Accuracy for majority:** $TPR = \frac{TP}{TP+FP}$**.**
    - **Accuracy for minority:** $TNR = \frac{TN}{TN+FN}$**.**
  - **Issue:** ω **must be specified prior to the evolutionary search.**

- **Our approach: Multi-objective GP (MOGP) using Pareto-based fitness schemes.**
  - **Two fitness functions:** TPR **and** TNR**.**
  - **Dominance measures (use one of them):**
    - **SPEA2 (dominance rank and dominance count) and**
    - **NSGAII (dominance rank only).**
  - **Trick: Crowding distance measure to encourage diverse solutions on frontier.**

Outline of This Lecture

Research Question I

Question I: Do the designed MOGP approach with Pareto-based fitness schemes outperform the SOGP?

$\Longrightarrow$ **Evaluation of MOGP fitness schemes and comparison between MOGP and SOGP on evolving diverse-classifier.**

Research Question II

Question II: Are our MOGP methods domain specific?

- **Six selected benchmark data sets:**
  - From different problem domains.
  - Varying levels of class imbalance.
  - Different complexities where some tasks are easily separable (e.g., Yst_2).
  - Well- vs. sparsely represented.
  - High vs. low dimensionality.
  - Binary vs. real-valued feature types.

Name	Classes (Minority/Majority)	Number of Examples: Total	Minority	Majority	Imb. Ratio	Features: No.	Type
Ion	Good/bad (ionosphere radar signal)	351	126 (35.8%)	225 (64.2%)	1:3	34	Real
Spt	Abnormal/normal (cardiac tomography scan)	267	55 (20.6%)	212 (79.4%)	1:4	22	Binary
Ped	Pedestrian/background (image cut-out)	24 800	4800 (19.4%)	20 000 (80.6%)	1:4	22	Real
Yst$_1$	*mit*/nontarget (protein sequence)	1482	244 (16.5%)	1238 (83.5%)	1:6	8	Real
Yst$_2$	*me3*/nontarget (protein sequence)	1482	163 (10.9%)	1319 (89.1%)	1:9	8	Real
Bal	Balanced/unbalanced (balance scale)	625	49 (7.8%)	576 (92.2%)	1:12	4	Integer

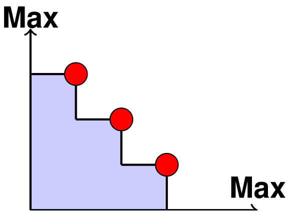
Figure 6: Unbalanced classification tasks used in the experiments. Table II of [1]. For each task, 50% of the examples in each class were randomly chosen for the training and the test sets. $\rightarrow$ Preserve the same class imbalance ratio.

Research Question III

Question III: What quality indicators should be used?

Quality Indicator: Hyperarea (Hypervolume, HV)

- **Hyperarea, (also known as the hypervolume) of the evolved Pareto-approximated fronts.**
  - **Classic performance indicator !**
  - **Area dominated by the Pareto-approximated solutions *in the objective space*.**
  - $=$ **sum of the areas of individual trapezoids fitted under each front solution *in the objective space*.**



Take Home Message

- SPEA2's average hyperarea is statistically better than NSGAII on the three tasks, and not statistically different to NSGAII on the remaining three tasks.
- The hyperarea of the Pareto-optimal (PO) front is also better in SPEA2 for all tasks except *Bal*.
- The two MOGP approaches show similar average training times.
  ← Quality and time should both be considered!

Task	NSGAII Fitness			SPEA2 Fitness		
	Hyperarea		Training Time	Hyperarea		Training Time
	Average	PO Front		Average	PO Front	
Ion	0.793 ± 0.041	0.952	8.3 s ± 1.3	**0.848 ± 0.041**	**0.992**	9.3 s ± 2.4
Spt	0.733 ± 0.026	0.938	16.9 s ± 2.1	0.732 ± 0.032	**0.971**	9.7 s ± 2.5
Ped	0.881 ± 0.013	0.903	3.5 m ± 52.6	**0.902 ± 0.019**	**0.922**	3.9 m ± 1.1
Yst_1	0.793 ± 0.008	0.917	23.5 s ± 4.5	0.793 ± 0.009	**0.931**	20.8 s ± 7.1
Yst_2	0.942 ± 0.008	0.986	23.5 s ± 4.4	**0.949 ± 0.011**	**0.991**	20.1 s ± 8.1
Bal	0.749 ± 0.049	**0.993**	20.1 s ± 2.6	0.757 ± 0.063	0.985	15.2 s ± 3.9

Figure 7: Average Hyperarea of evolved Pareto-approximated fronts, Pareto-optimal (PO) fronts and training times for the MOGP over 50 independent trials. Table III of [1]. PO front is the set of nondominated solutions from the union of all Pareto-approximated fronts evolved from the 50 independent runs.

Observation: **"SPEA2's average hyperarea is *not statistically different* to NSGAII on the three tasks: *Spt*, *Yst*$_1$ and *Bal*."**

Task	NSGAII Fitness			SPEA2 Fitness		
	Hyperarea		Training Time	Hyperarea		Training Time
	Average	PO Front		Average	PO Front	
Ion	0.793 ± 0.041	0.952	8.3 s ± 1.3	**0.848 ± 0.041**	**0.992**	9.3 s ± 2.4
Spt	0.733 ± 0.026	0.938	16.9 s ± 2.1	0.732 ± 0.032	**0.971**	9.7 s ± 2.5
Ped	0.881 ± 0.013	0.903	3.5 m ± 52.6	**0.902 ± 0.019**	**0.922**	3.9 m ± 1.1
Yst$_1$	0.793 ± 0.008	0.917	23.5 s ± 4.5	0.793 ± 0.009	**0.931**	20.8 s ± 7.1
Yst$_2$	0.942 ± 0.008	0.986	23.5 s ± 4.4	**0.949 ± 0.011**	**0.991**	20.1 s ± 8.1
Bal	0.749 ± 0.049	**0.993**	20.1 s ± 2.6	0.757 ± 0.063	0.985	15.2 s ± 3.9

Question IV: What do their Pareto fronts look like? Can we approximate the fronts from independent runs?

Outline of This Lecture

Remark: MOGP outputs the evolved Pareto approximate front, consisting of diverse classifiers.
$\leftarrow$ Hey! This is what an ensemble needs!

Question V: Can we evolve ensembles using MOGP?

Answer(?): Sounds possible. An ensemble can be represented as a tree! However . . .

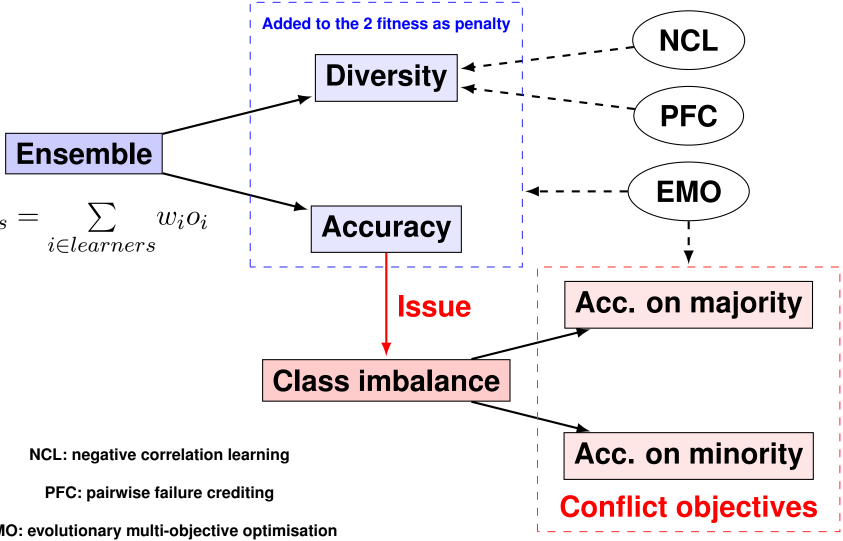
Question VI: Should we use all the evolved classifiers to build an ensemble or some of them?

Follow-up Question: If using some of them, how to select?

$\Longrightarrow$ Urvesh Bhowan et al. "Reusing genetic programming for ensemble selection in classification of unbalanced data". In: *IEEE Transactions on Evolutionary Computation* 18.6 (2014), pp. 893–908

Recall

Diversity and Accuracy as Two Conflicting Objectives



$$o_{ens} = \sum_{i \in learners} w_i o_i$$

NCL: negative correlation learning

PFC: pairwise failure crediting

EMO: evolutionary multi-objective optimisation

Overview of MOGP and Ensemble Selection Process

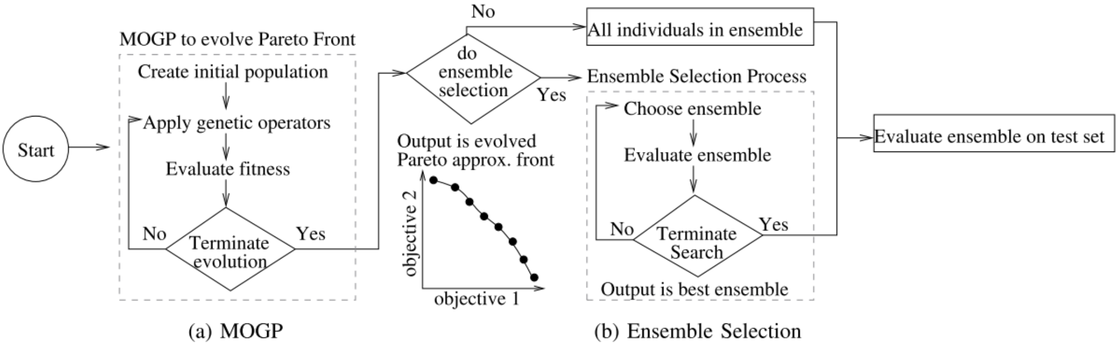


Figure 8: Figure 1 of [1].

Two Objectives and Fitness Design

- **Each class: Accuracy + Pairwise failure crediting (PFC):**

$$f_c(\mathbf{x}_i) = \frac{1 - Err(c, \mathbf{x}_i)}{N_c} + PFC_{c,i},$$

where

- $PFC_{c,i} = \frac{1}{|Pop|-1} \sum_{j=1, j \neq i}^{|Pop|} \frac{HD_c(\mathbf{x}_i, \mathbf{x}_j)}{Err(c, \mathbf{x}_i) + Err(c, \mathbf{x}_j)}$.
- $Err(c, \mathbf{x}_i)$ **is the number of errors made by individual** $\mathbf{x}_i$ **on class** c**.**
- N_c **is the number of training examples in class** c**.**
- Pop **is the population.**
- HD **is the Hamming distance between the outputs of two individuals** $\mathbf{x}_i$ **and** $\mathbf{x}_j$ **on the examples from class** c**. So** HD **is the number of outputs where the predicted class labels are different between two solutions.**
- $PFC_{c,i}$ **represents the diversity of individual** $\mathbf{x}_i$ **on class** c**, and** PFC **values range between 0 and 1 where higher** PFC **values mean better diversity.**

Why PFC?

- **Why not using Negative Correlation Learning (NCL)?**
  $\rightarrow$ PFC is a population-based measure, whereas other measures such as NCL evaluate diversity relative to the ensemble's output.
  $\rightarrow$ Require that the ensemble members are known a priori. $\rightarrow$ Conflict to our aim.

- **[1] found that classifier diversity was better using PFC in fitness compared to the widely-used NCL.**

Pareto Dominance Ranking

- **Pareto fitness using SPEA2:**

$$fitness(\mathbf{x}_i) = \sum_{j=1, \mathbf{x}_j \succ \mathbf{x}_i}^{pop} Strength(\mathbf{x}_j),$$

where $Strength(\mathbf{x}_i) = |\{j | j \in Pop \wedge \mathbf{x}_i \succ \mathbf{x}_j\}|$**, which indicates the number of solutions it dominates. The fitness is determined by the strength of individual's dominators.**

Why SPEA2?

- **SPEA2 uses both dominance count (i.e. number of others that a given individual dominates) and dominance rank (i.e. number of others that dominate a given individual) in the fitness calculation.**
- **Dominance rank tends to reward exploration at the edges of the frontier, while dominance count rewards exploitation in the middle of frontier.**
- **MOGP with SPEA2 is better at pushing the Pareto front outwards toward good performance on all objectives compared to NSGAII, which found a wider spread of individuals along the whole of the frontier [1].**

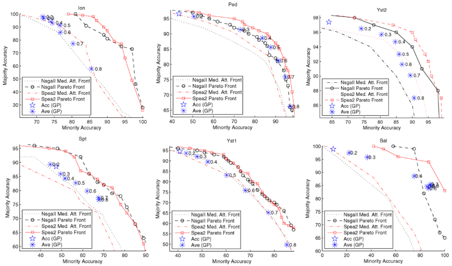


Figure 9: The average evolved front and Pareto-optimal fronts. Figure 3 of [1].

Forming the GP Ensemble I

- **Ensemble Representation:**
  - **An (optimised) ensemble $\rightarrow$ a GP composite solution $\rightarrow$ a single genetic program which links to multiple evolved Pareto-approximated front classifiers (terminal nodes).**
  - **Terminal set: $\{\emptyset, p_1, \ldots, p_T\}$, where p_i refers to a link to the i^{th} base classifier from the Pareto-approximated front (of size T) from a given MOGP run.**

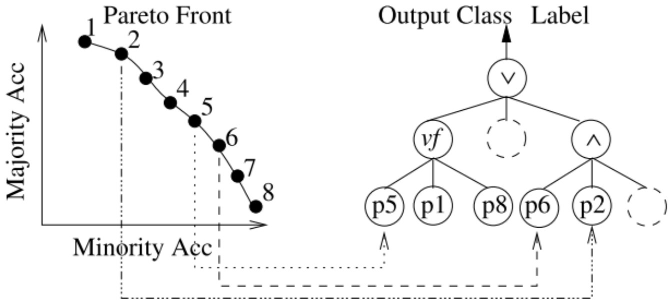


Figure 10: Figure 2 of [2]. Non-terminal node: an aggregation strategy.

Forming the GP Ensemble II

- **Aggregation strategies (non-terminal nodes, function sets).**
  1. ***Composite Voting Solutions (CSVote)*: Traditional majority vote (MVS) operator (vt).**
     - **When the function vt is the root node, it computes the MVS of each base classifier within the tree.**
     - **When vt is an internal (non-leaf) node in a CSVote tree, vt serves no purpose other than to join terminal (leaf) nodes or other vt nodes to the root node.**
  2. ***Composite Logical Solutions (CSLogic)*: logical operators.**
     - **Function set: $\{\vee, \wedge, vf\}$, where vf indicate majority voting.**
     - **Greater "decisions making" abilities than traditional MVS.**

Question: Whether the logical operators are better for ensemble generalisation compared to MVS?

Benchmark Data Sets

Name	Description	Total	Minority Class Num	%	IR	Feat. # Type
Ion	Ionosphere radar signal [50]	351	126	35.8%	1:3	34 $\mathbb{R}$
Spt	Tomography scan [50]	267	55	20.6%	1:4	22 B
Ped	Pedestrian Image [51]	24800	4800	19.4%	1:4	22 $\mathbb{R}$
Eco$_1$	Ecoli protein* (*eim*) [50]	336	77	22.9%	1:4	7 $\mathbb{R}$
Eco$_2$	Ecoli protein* (*epp*) [50]	336	52	15.5%	1:6	7 $\mathbb{R}$
Yst$_1$	Yeast protein* (*mit*) [50]	1482	244	16.5%	1:6	8 $\mathbb{R}$
Yst$_2$	Yeast protein* (*me3*) [50]	1482	163	10.9%	1:9	8 $\mathbb{R}$
Vow	Vowel* (*class0*) [32]	988	90	9.1%	1:10	13 $\mathbb{R}$
Led	LED display* (*class1*) [32]	443	37	8.4%	1:11	7 $\mathbb{R}$
Bal	Balance scale* (*classB*) [50]	625	49	7.8%	1:12	4 $\mathbb{Z}$
Aba	Abalone* (9 *v* 18) [32]	731	42	5.7%	1:17	8 $\mathbb{R}$
Shut	Statlog Shuttle* (c2 *v* c4) [32]	129	6	4.7%	1:20	9 $\mathbb{Z}$

Figure 11: Table I of [1].

Parameter Settings

- **Termination criteria:**
  - **maximum 30 generations or**
  - **when as a composite solution with 100% accuracy on both classes on the training set is evolved.**

- **Parameter settings:**

	GP
Initialisation	Ramped half-and-half
Population size	300
D_{max}	2 and 3
Crossover rate	0.6
Mutation rate	0.35
Tournament size	7
Elitism rate	0.05

- **#Independent trials**$= 50$.

Baselines

- **FULL: The full ensembles without ensemble selection where all non-dominate front members (for a given MOGP run) form the ensemble.**
- **off-EEL: Offline evolutionary ensemble learning [5].**
  1. **Run a standard evolutionary learning algorithm, use the final population as pool of classifiers.**
  2. **Ensemble selection based on the margin-based criterion.**
     - **Initialise the classifier ensemble L to the classifier h^* that has the smallest error rate on the training set in the population, thus $L = \{h^*\}$.**
     - $Margin_i = |h_j(x_i) = y_i, h_j \in L| - |h_j(x_i) = y_i', h_j \in L|$.
     - **The more negative the margin, the more classifiers need to be added to the ensemble in order to correctly classify x_i.**

Representative Experimental Results I

Task	Approach	Average Ensemble Sizes	Test Set							Training Set
			Geomean Accuracy %		Statistical Significance			Class Accuracy %		Geomean Accuracy %
			Best	Average	Rank	Beats	p-value	Minority	Majority	
Ion	**CSVote$_2$**	8.9 ± 0.4	95.5	$\mathbf{90.1 \pm 2.2}$	1	$\{3, 4\}$	p=5.1×10^{-31}	85.2 ± 4.1	95.5 ± 2.6	96.5 ± 1.8
	off-EEL	21.2 ± 7.4	95.9	89.8 ± 2.9	1	$\{3, 4\}$		83.6 ± 5.3	96.6 ± 2.8	98.4 ± 0.9
	FULL	28.1 ± 4.5	94.7	88.4 ± 3.0	2	$\{4\}$		84.9 ± 5.1	92.4 ± 6.5	97.5 ± 1.2
	CSVote$_3$	21.6 ± 6.3	94.6	86.6 ± 3.5	3	$\{4\}$		81.9 ± 5.4	91.9 ± 6.3	99.3 ± 0.6
	CSLogic$_2$	8.3 ± 1.0	93.0	86.2 ± 3.4	3	$\{4\}$		80.6 ± 5.8	92.4 ± 4.5	99.5 ± 0.5
	CSLogic$_3$	26.5 ± 9.2	88.8	60.9 ± 17.0	4	$\{\}$		67.6 ± 30.3	70.7 ± 32.6	99.4 ± 0.5
Spt	**off-EEL**	10.7 ± 5.2	79.0	$\mathbf{72.4 \pm 3.0}$	1	$\{2-4\}$	p=8.2×10^{-36}	66.3 ± 8.6	79.9 ± 6.7	94.7 ± 1.3
	CSVote$_2$	8.7 ± 0.8	78.5	72.2 ± 2.6	1	$\{2-4\}$		64.6 ± 6.3	81.0 ± 4.9	92.8 ± 3.0
	CSVote$_3$	17.7 ± 7.8	76.6	68.8 ± 3.5	2	$\{4\}$		55.7 ± 7.4	85.7 ± 4.0	96.0 ± 0.5
	CSLogic$_2$	9.0 ± 0.1	74.8	67.3 ± 3.1	3	$\{4\}$		54.9 ± 6.0	82.9 ± 4.0	95.3 ± 0.7
	FULL	27.3 ± 4.0	71.3	63.5 ± 3.7	4	$\{\}$		44.6 ± 5.5	90.8 ± 2.4	91.9 ± 2.5
	CSLogic$_3$	23.3 ± 10.8	72.7	43.7 ± 28.3	4	$\{\}$		41.1 ± 32.2	81.2 ± 23.4	96.4 ± 0.4
Ped	**CSVote$_2$**	8.7 ± 0.1	92.0	$\mathbf{89.4 \pm 2.1}$	1	$\{3\}$	p=1.2×10^{-8}	90.7 ± 2.3	88.1 ± 2.4	93.3 ± 2.6
	CSVote$_3$	22.7 ± 1.8	91.7	89.2 ± 2.2	1	$\{3\}$		88.1 ± 2.2	90.4 ± 3.0	91.8 ± 1.7
	off-EEL	55.2 ± 5.0	91.7	89.2 ± 1.5	1	$\{3\}$		90.6 ± 1.5	87.9 ± 1.4	91.3 ± 1.6
	CSLogic$_2$	9.0 ± 0.0	91.3	88.3 ± 2.4	1	$\{3\}$		87.8 ± 1.6	88.8 ± 3.1	88.0 ± 2.1
	FULL	71.6 ± 10.2	91.2	87.1 ± 2.6	2	$\{\}$		82.4 ± 4.6	92.1 ± 2.5	86.7 ± 1.9
	CSLogic$_3$	24.0 ± 3.5	88.5	85.9 ± 2.4	3	$\{\}$		85.0 ± 2.1	86.9 ± 2.4	88.1 ± 2.6
Eco$_1$	**CSVote$_2$**	7.6 ± 1.2	82.0	$\mathbf{77.8 \pm 2.9}$	1	$\{3-6\}$	p=2.0×10^{-38}	91.5 ± 5.0	66.2 ± 4.1	98.8 ± 1.2
	CSVote$_3$	18.3 ± 9.1	82.9	77.1 ± 4.0	2	$\{4-6\}$		93.7 ± 3.9	63.7 ± 6.2	99.2 ± 0.9
	off-EEL	7.9 ± 2.5	83.3	75.0 ± 4.4	3	$\{6\}$		90.9 ± 5.0	62.1 ± 7.0	99.9 ± 0.2
	CSLogic$_2$	8.4 ± 1.2	80.6	72.5 ± 7.0	4	$\{6\}$		88.2 ± 7.0	60.2 ± 10.1	99.9 ± 0.2
	FULL	8.3 ± 1.8	81.0	71.7 ± 6.0	5	$\{\}$		91.5 ± 5.1	56.8 ± 9.9	99.5 ± 0.5
	CSLogic$_3$	12.6 ± 4.6	82.2	60.9 ± 16.3	6	$\{\}$		82.9 ± 18.3	52.2 ± 26.1	99.9 ± 0.2
Eco$_2$	**CSVote$_2$**	8.6 ± 1.1	100.0	$\mathbf{99.9 \pm 0.3}$	1	$\{3, 4\}$	p=8.9×10^{-17}	99.9 ± 0.5	99.8 ± 0.4	98.8 ± 1.6
	CSVote$_3$	23.7 ± 6.5	100.0	99.8 ± 0.5	1	$\{3, 4\}$		99.8 ± 0.8	99.7 ± 0.6	98.5 ± 1.5
	off-EEL	10.6 ± 3.8	100.0	99.8 ± 0.4	1	$\{3, 4\}$		99.9 ± 0.5	99.6 ± 0.5	99.8 ± 0.4
	CSLogic$_2$	7.1 ± 1.5	100.0	99.4 ± 0.8	2	$\{4\}$		99.5 ± 1.3	99.3 ± 1.1	99.9 ± 0.2
	FULL	15.4 ± 2.7	100.0	98.8 ± 1.5	3	$\{4\}$		97.9 ± 3.1	99.6 ± 0.6	98.8 ± 1.5
	CSLogic$_3$	16.2 ± 4.2	98.1	84.2 ± 15.1	4	$\{\}$		80.6 ± 23.7	92.5 ± 12.5	99.9 ± 0.2
Yst$_1$	**off-EEL**	29.2 ± 9.3	77.6	$\mathbf{74.4 \pm 1.1}$	1	$\{3-4\}$	p=9.6×10^{-17}	70.6 ± 5.4	78.8 ± 5.6	85.3 ± 1.0
	CSVote$_2$	9.0 ± 0.0	76.9	72.9 ± 1.5	2	$\{4\}$		78.7 ± 5.8	66.6 ± 7.7	81.7 ± 4.2
	FULL	39.7 ± 5.1	75.6	72.1 ± 2.4	2	$\{4\}$		64.6 ± 4.8	82.5 ± 4.3	83.9 ± 1.1
	CSVote$_3$	17.5 ± 6.3	75.0	72.5 ± 1.3	3	$\{4\}$		67.8 ± 5.1	77.9 ± 4.9	86.6 ± 0.8
	CSLogic$_2$	8.9 ± 0.3	75.8	72.5 ± 1.7	3	$\{4\}$		64.6 ± 3.9	81.4 ± 3.1	87.8 ± 0.9
	CSLogic$_3$	28.9 ± 8.9	73.2	47.6 ± 17.6	4	$\{\}$		56.5 ± 36.8	65.9 ± 34.3	87.7 ± 0.9

Figure 12: Table II of [1]. An ANOVA F-test [52] is used to test the null hypothesis, i.e., no significant difference between the approaches over 50 runs (*p-value*). A post-hoc multiple comparisons test using Tukey's Honestly Significant Difference (HSD) is used to determine the statistically significant differences between group means (*Rank* and *Beats*).

Representative Experimental Results II

Take Home Message

- Ensemble selection is important: poorer performance by FULL vs. CSVote_2 and off-EEL.
  $\rightarrow$ FULL contains more members which have a stronger majority class bias.
- Ensemble selection can successfully exclude biased individuals from the ensemble.
  $\rightarrow$ Improve ensemble accuracy on the important minority class.
- Training performance for the ensemble selection approaches is good (achieving near-perfect accuracy) in nearly all tasks.
- The composite solutions, in particular CSVote, may be particularly useful in optimisation problems or online learning which does not use an unseen test set.

Outline of This Lecture

Receiver Operating Characteristic (ROC) Graphs

- **A ROC curve plots the TPR vs. the FPR as a discriminative threshold on the confidence of an instance being positive is varied.**
- **Area under the ROC curve (AUC): a fair indicator to measure the classifier performance for binary classification.**
- **ROC convex hull (ROCCH): the least convex majorant (LCM) of the empirical ROC curve and covers potential optima for a given set of classifiers.**

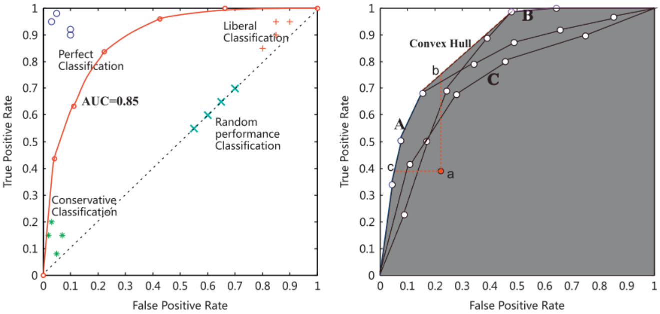
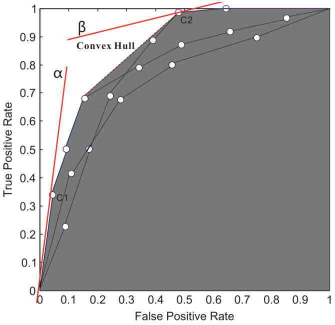
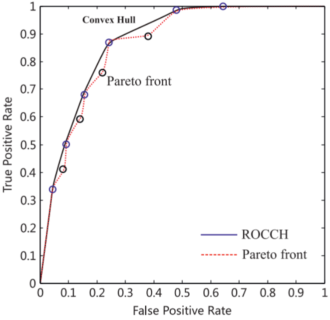


Figure 13: ROC Graph and ROCCH. Figures 1 and 3 of [9].

ROCCH to Search Optimal Points

- **A classifier is potentially optimal iff it touches the ROCCH.**
- **Moving the iso-performance line (red curves on left Figure) until it gets in contact with a point in ROCCH, the joint point with a larger TPR-intercept represents a classifier which is potentially optimal.**
- **Similar to the graphical method for solving linear programming problems.**





Motivation

- **A classifier is potentially optimal iff it touches the ROCCH.**
- **Search a group of classifiers to**
  1. **maximise ROCCH performance $\rightarrow$ 2 conflicting objectives**
     1.1 **minimise FPR**
     1.2 **maximise TPR**
  2. **Performance indicator: AUC in binary classification problems.**

$\rightarrow$ **MOGP can handle this multi-objective optimisation problem!**

Research Questions:

1. **Will GP approaches outperform traditional ML algorithms on maximising ROCCH performance in classification tasks?**

2. **Will MOGP approaches outperform SOGP approaches ?**

3. **Will GP approaches benefit from local search?**

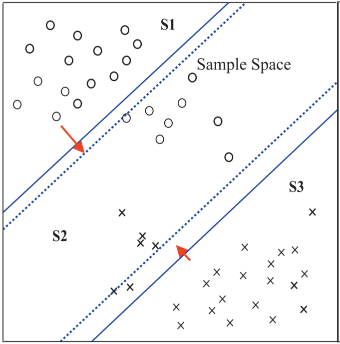
Additional Ingredient: Local Search for GP I

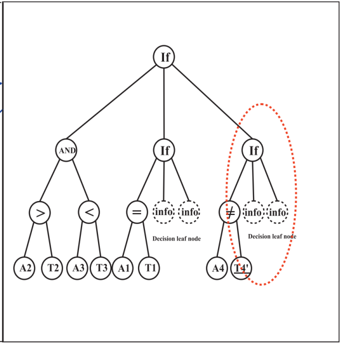
- **Shifting operator**:
  - **Shift the hyperplane $\rightarrow$ threshold adjusting in genetic decision tree (GDT).**
  - **Information gain for measuring the quality of a classifier x:**

$$E(x) = \frac{\sum_{\forall leaves\ l \in x} (1 + \sum_{k=1}^{2} p(l,k) \log_2 p(l,k))(\sum_{k=1}^{2} P(l)[k] - 1)}{|All\ instances| - |leaves \in x|},$$

    where
    - $P(l)[k]$ **and** $p(l,k)$ **are the number and the probability of instances with label** k **in the** l**th decision node,**
    - $p(l,k) = \frac{P(l)[k]}{\sum_{i=1}^{2} P(l)[i]}$.
  - **It does not always work well.**

Additional Ingredient: Local Search for GP II





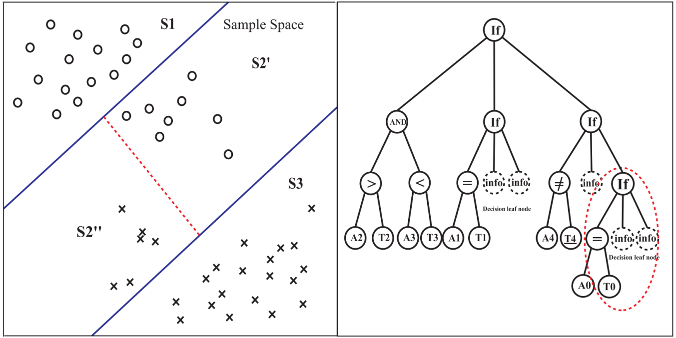
Additional Ingredient: Local Search for GP III

- **Splitting operator**:
  - **One subspace $\mathcal{S}'$ which is not pure (e.g., information gain < 0.1) will be considered for the splitting operator with a probability $= \frac{|Instances \in \mathcal{S}'|}{|All\ instances|}$.**
  - **Information gain for measuring the quality of subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$:**

$$InfoGain(\mathcal{S}) = \frac{p_1 + n_1}{n + p} Info(\mathcal{S}_1) + \frac{n + p - p_1 - n_1}{n + p} Info(\mathcal{S}_2),$$

where $Info(\mathcal{S}) = -(\frac{p}{n+p} \log_2 \frac{p}{n+p} + \frac{n}{n+p} \log_2 \frac{n}{n+p})$, supposing $\mathcal{S}$ has p positive instances and n negative instances and the subspace $\mathcal{S}_1$ has p_1 positive instances and n_1 negative instances.

Additional Ingredient: Local Search for GP IV



Nondominated Sorting GP (NSGP)

Algorithm 4. *NSGP-II*(P,Max,N).

Require: $Max \geq 0 \vee P = null \vee N > 0$
Max is the maximum evaluations
P is the population
N is the population size
Ensure: NSGA-II
Let $m = 0, t = 0$
Initialize the population P_t by ramped-half-and-half method
Evaluate each individual in P_t and $m = m + N$
while $m \leq Max$ **do**
  Generate offspring Q_t from P_t by tree-based crossover
  Shifting operator with probability p_{sf}
  Splitting operator with probability p_{sp}
  Evaluate each changed offspring in Q_t
  $m = m + |$ changed-offspring $|$
  $R_t = P_t \cup Q_t$
  $\mathbf{F}$=fast- nondominated-sort (R_t)
  $P_{t+1} = \emptyset$ and $i=0$
  while $|P_{t+1}| + |\mathbf{F}_i| \leq N$ **do**
    crowding-distance-assignment($\mathbf{F}_i$)
    $P_{t+1} = P_{t+1} \cup \mathbf{F}_i$
    $i = i + 1$

Experimental Design I

- **Approaches:**
  - **SOGP (S-*): Nondominated Sorting GP (NSGP).**
  - **MOGP, SMS-MOGP (selection based on the hypervolume measure + nondominated sorting) and AG-MOGP (approximation-guided).**
- **Baselines: Some traditional ML algorithms.**
- **27 balanced and imbalanced benchmark data sets of two-class problems.**
- **5-fold cross-validation for 20 times.**
- **Generation number M:**
  - **The train part *Train* of the 5-fold cross-validation on *Data* is taken and *Alg* is applied to *Train* by 5-fold cross-validation for one time.**
  - **Then set M to the generation index at which *Alg* had the best performance.**
- **Terminals of GP: 0 and 1 (negative and positive).**
- **Every classifier is constructed as `if-then-else` tree which involves `and`, `or`, `not`, $<$, $>$ and $=$ as operator symbols.**

Experimental Design II

	Objective	Maximize ROCCH	
Terminals of GP	{0,1} with 1 representing "Positive"; 0 representing "Negative"	Function set of GP	If–then–else, and, or, not, $>$, $<$, $=$.
Data sets	27 UCI data sets	Algorithms	15 algorithms in Table 2
Crossover rate	0.9	Mutation rate	0.1
Shifting rate	0.1	Splitting rate	0.1
Parameters for GP	P (population size)$=$100; G (maximum evaluation times)$=$M Number of runs: 5-fold cross-validation 20 times	Termination criterion	Maximum of G of evaluation time has been reached
Selection strategy	Tournament selection, Size$=$4	Max depth of initial/inprocess individual program	3/17

Figure 14: Parameter setting. Table 3 of [9].

Experimental Results

Local Search Works Well!

Data set	S-NSGP-II	NSGP-II	S-MOGP/D	MOGP/D
australian	90.93 ± 2.52	$\mathbf{92.00 \pm 2.46}$	88.09 ± 5.37	$\mathbf{91.68 \pm 2.44}$
bands	71.71 ± 5.43	$\mathbf{77.70 \pm 3.49}$	69.05 ± 4.47	$\mathbf{76.47 \pm 4.05}$
bcw	98.12 ± 0.80	98.19 ± 0.99	97.71 ± 1.33	$\mathbf{98.07 \pm 1.13}$
crx	90.18 ± 3.12	$\mathbf{91.79 \pm 2.47}$	89.53 ± 4.88	$\mathbf{91.58 \pm 2.32}$
euthyroid	79.27 ± 9.20	$\mathbf{96.78 \pm 1.37}$	72.46 ± 10.39	$\mathbf{94.47 \pm 6.91}$
german	73.00 ± 3.94	$\mathbf{74.03 \pm 2.81}$	68.08 ± 5.35	$\mathbf{73.52 \pm 2.97}$
haberman	65.55 ± 6.60	$\mathbf{67.08 \pm 6.19}$	63.68 ± 7.17	$\mathbf{66.60 \pm 6.58}$
hill-valley	50.30 ± 1.47	$\mathbf{53.19 \pm 2.61}$	50.07 ± 1.47	$\mathbf{53.02 \pm 2.59}$
house-votes	97.01 ± 3.82	98.10 ± 1.39	96.50 ± 2.87	$\mathbf{97.84 \pm 1.46}$
hypothyroid	79.63 ± 11.60	$\mathbf{97.99 \pm 1.52}$	77.41 ± 14.60	$\mathbf{97.11 \pm 2.06}$
ionosphere	86.81 ± 6.76	$\mathbf{91.83 \pm 3.98}$	84.61 ± 6.73	$\mathbf{91.42 \pm 3.56}$
kr-vs-kp	88.67 ± 7.32	$\mathbf{98.01 \pm 0.85}$	80.41 ± 8.04	$\mathbf{98.12 \pm 0.99}$
mammographic	89.08 ± 2.05	89.79 ± 1.80	87.71 ± 2.52	$\mathbf{89.45 \pm 2.00}$
monks-1	94.80 ± 3.43	$\mathbf{99.93 \pm 0.53}$	88.75 ± 11.39	$\mathbf{99.45 \pm 1.97}$
monks-2	77.65 ± 9.50	$\mathbf{93.60 \pm 5.25}$	68.18 ± 10.74	$\mathbf{89.82 \pm 16.76}$
monks-3	98.22 ± 4.26	$\mathbf{100.00 \pm 0.00}$	94.51 ± 9.50	$\mathbf{99.84 \pm 0.45}$
mushroom	98.70 ± 1.61	$\mathbf{99.95 \pm 0.10}$	96.93 ± 3.15	$\mathbf{99.77 \pm 0.30}$
parkinsons	85.09 ± 6.58	86.17 ± 5.96	80.87 ± 8.00	$\mathbf{86.96 \pm 5.02}$
pima	77.22 ± 3.52	$\mathbf{80.61 \pm 3.21}$	72.54 ± 5.07	$\mathbf{80.35 \pm 2.86}$
sonar	70.42 ± 6.01	$\mathbf{80.09 \pm 5.55}$	67.51 ± 7.43	$\mathbf{79.68 \pm 6.05}$
spambase	70.97 ± 8.55	$\mathbf{96.36 \pm 0.57}$	64.17 ± 7.67	$\mathbf{95.80 \pm 0.60}$
spect	75.47 ± 5.05	76.52 ± 6.91	73.90 ± 8.34	$\mathbf{76.97 \pm 7.85}$
spectf	68.30 ± 5.95	$\mathbf{73.38 \pm 5.55}$	66.43 ± 8.58	$\mathbf{73.58 \pm 5.65}$
tic-tac-toe	73.39 ± 8.99	$\mathbf{86.19 \pm 11.46}$	67.52 ± 11.04	$\mathbf{84.18 \pm 9.06}$
transfusion	68.97 ± 4.89	$\mathbf{72.12 \pm 4.44}$	64.94 ± 4.75	$\mathbf{71.88 \pm 4.63}$
wdbc	93.52 ± 4.95	$\mathbf{97.28 \pm 1.49}$	92.42 ± 4.73	$\mathbf{97.02 \pm 1.63}$
wpbc	59.52 ± 8.15	$\mathbf{67.41 \pm 8.33}$	59.42 ± 7.76	$\mathbf{66.61 \pm 7.41}$
Win–draw–loss	0–5–22	22–5–0	0–0–27	27–0–0

Experimental Results

Beat Some Traditional ML Algorithms!

Data set	EGP	FGP	GGP
australian	90.05 ± 3.06	85.56 ± 4.87	85.54 ± 3.83
bands	70.04 ± 5.05	53.99 ± 5.56	64.88 ± 4.89
bcw	97.35 ± 1.37	93.73 ± 2.11	93.85 ± 2.45
crx	90.68 ± 2.49	85.91 ± 3.57	86.36 ± 3.32
euthyroid	93.37 ± 5.81	50.01 ± 0.11	79.41 ± 13.12
german	70.81 ± 3.42	51.75 ± 3.51	67.14 ± 5.36
haberman	62.97 ± 7.63	50.66 ± 4.25	63.98 ± 6.68
hill-valley	50.18 ± 2.15	50.09 ± 1.39	49.90 ± 3.25
house-votes	97.75 ± 1.63	94.63 ± 4.00	95.23 ± 3.85
hypothyroid	96.55 ± 2.55	52.35 ± 3.27	93.45 ± 5.91
ionosphere	87.22 ± 5.84	80.87 ± 7.60	79.86 ± 7.01
kr-vs-kp	85.71 ± 6.65	62.16 ± 8.52	71.89 ± 6.02
mammographic	88.96 ± 1.97	82.76 ± 3.60	84.73 ± 3.46
monks-1	85.96 ± 11.96	51.21 ± 9.96	75.03 ± 5.25
monks-2	80.48 ± 12.05	50.01 ± 6.29	53.28 ± 6.92
monks-3	99.59 ± 0.49	87.48 ± 10.72	86.75 ± 9.04
mushroom	98.68 ± 1.88	84.67 ± 8.24	89.44 ± 4.47
parkinsons	81.92 ± 7.80	76.62 ± 8.22	75.97 ± 7.19
pima	76.27 ± 4.94	50.88 ± 1.29	70.73 ± 3.44
sonar	73.33 ± 7.01	54.17 ± 6.81	68.22 ± 7.38
spambase	85.28 ± 5.53	76.14 ± 7.16	76.58 ± 4.30
spect	74.36 ± 7.01	68.21 ± 10.68	71.99 ± 7.18
spectf	71.76 ± 7.04	58.69 ± 9.06	69.16 ± 7.16
tic-tac-toe	71.89 ± 12.11	63.35 ± 9.73	63.35 ± 10.15
transfusion	71.31 ± 5.21	50.48 ± 0.89	67.46 ± 4.37
wdbc	95.12 ± 2.92	87.25 ± 4.54	90.39 ± 2.83
wpbc	66.83 ± 9.90	56.47 ± 7.41	60.15 ± 8.92
NSGP-II	23–4–0	27–0–0	27–0–0
MOGP/D	22–5–0	27–0–0	27–0–0

Experimental Results

NSGP-II is the best algorithm!

Wilcoxon rank-sum test results for MOGP methods with and without local search.

Algorithms	NSGP-II	MOGP/D	SMS-MOGP	AG-MOGP
NSGP-II	–	9–18–0	9–16–2	6–20–1
MOGP/D	0–18–9	–	3–20–4	4–20–3
SMS-MOGP	2–16–9	3–20–4	–	2–23–2
AG-MOGP	1–20–6	3–20–4	2–23–2	–
Algorithms	S-AGE-MOGP	S-MOGP/D	S-SMS-MOGP	S-NSGP-II
S-AG-MOGP	–	22–5–0	19–8–0	5–20–2
S-MOGP/D	0–5–22	–	3–18–6	0–8–19
S-SMS-MOGP	0–8–19	6–18–3	–	0–9–18
S-NSGP-II	2–20–5	19–8–0	18–9–0	–

Summary

- **GP is good for evolving the syntax of arithmetic expressions, formulas in first-order predicate logic, programs or learners.**
- **GP can not only be used to evolve single learners (functions), but also ensembles.**
- **Ensemble generation and ensemble selection using GP do not require configuring the number of base classifiers or ensemble aggregation strategy a priori.**
- **MOGP is helpful for evolving classifiers that handle class imbalance.**
- **Local search can enhance GP /MOGP.**
- **Runtime is a problem!**
  - **GP-based algorithms need much more time than traditional ML algorithms.**
  - **MOGP methods with local search consumes more time than their counterparts without local search.**

References for This Lecture I

[1] Urvesh Bhowan et al. "Evolving diverse ensembles using genetic programming for classification with unbalanced data". In: *IEEE Transactions on Evolutionary Computation* 17.3 (2013), pp. 368–386.

[2] Urvesh Bhowan et al. "Reusing genetic programming for ensemble selection in classification of unbalanced data". In: *IEEE Transactions on Evolutionary Computation* 18.6 (2014), pp. 893–908.

[3] Hugo De Garis. "Genetic programming: Building artificial nervous systems using genetically programmed neural network modules". In: *Machine Learning Proceedings 1990*. Elsevier, 1990, pp. 132–139.

[4] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Vol. 53. Springer, 2003.

[5] Christian Gagné et al. "Ensemble learning for free with evolutionary algorithms?" In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM. 2007, pp. 1782–1789.

References for This Lecture II

[6] John R Koza. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Vol. 34. Stanford University, Department of Computer Science Stanford, CA, 1990.

[7] John R Koza. *Genetic programming: on the programming of computers by means of natural selection*. Vol. 1. MIT press, 1992.

[8] John R Koza, Forrest H Bennett III, and David Andre. *Method and apparatus for automated design of complex structures using genetic programming*. US Patent 5,867,397. 1999.

[9] Pu Wang et al. “Multiobjective genetic programming for maximizing ROC performance”. In: *Neurocomputing* 125 (2014), pp. 102–118.

[10] M-J Willis et al. “Genetic programming: An introduction and survey of applications”. In: *Second international conference on genetic algorithms in engineering systems: innovations and applications*. IET. 1997, pp. 314–319.

Reading List for Next Lecture

1. **R. J. Urbanowicz and J. H. Moore, "Learning classifier systems: A complete introduction, review, and roadmap," Journal of Artificial Evolution and Applications, vol. 2009, p. 1, 2009.**
2. **T. Kovacs, "Strength or accuracy: credit assignment in learning classifier systems," Springer Science & Business Media, 2012.**
3. **X. Yao and Md. M. Islam, "Evolving artificial neural network ensembles," IEEE Computational Intelligence Magazine, 3(1):31-42,February 2008.**
4. **X. Yao, "Evolving artificial neural networks,"** ***Proceedings of the IEEE*****, 87(9):1423-1447, September 1999.**
5. **X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks,"** ***IEEE Transactions on Neural Networks*****, 8(3):694-713, May 1997.**