On the Role of Genetic Algorithms in the Pattern Recognition Task of Classification

Isaac Sherman

Electrical Engineering and Computer Science

THE UNIVERSITY OF TENNESSEE KNOXVILLE

April 5, 2017

Outline

Outline

Basic Genetic Algorithm

```
BasicGA():
        Population := RandomInitialization()
        while True:
                for each Solution in Population:
                        Evaluate Solution
                        Assign Fitness to Solution
                newPopulation = SurviveAndBreed(Population)
                Population = newPopulation
        end while
```

Example

Consider an equation of the form

$$a\ op_1\ b\ op_2\ c\ op_3\ d = x$$

If we know x, we can use a GA to find operators and values which satisfy it. For instance,

$$3 \times 7 \times 3 + 5 = 68$$

Each digit can be encoded with 4 bits, each operator with 2. Values over 9 for digits are rerolled until valid.

Operators for Example

Bits	Operator
00	$+$
01	$-$
10	$\times$
11	$\div$

Therefore,

3	$\times$	7	$\times$	3	+	5
0011	11	1011	11	0011	00	0101

Example-> Fitness function

Here, we maximize $\frac{1}{|S_i - x|}$ and stop when $S_i = x$.
So, if x is 72, then the fitness of

$$3 \times 7 \times 3 + 5 = 68$$

is $\frac{1}{|68-72|} = 0.25$

Motivation

- Where do GAs fit in the greater scheme of pattern recognition?
- Given primitive mechanics, can they match or exceed theoretically-based methods?
- Can we build a generic, universal genetic algorithm for classification?

Outline

Outline

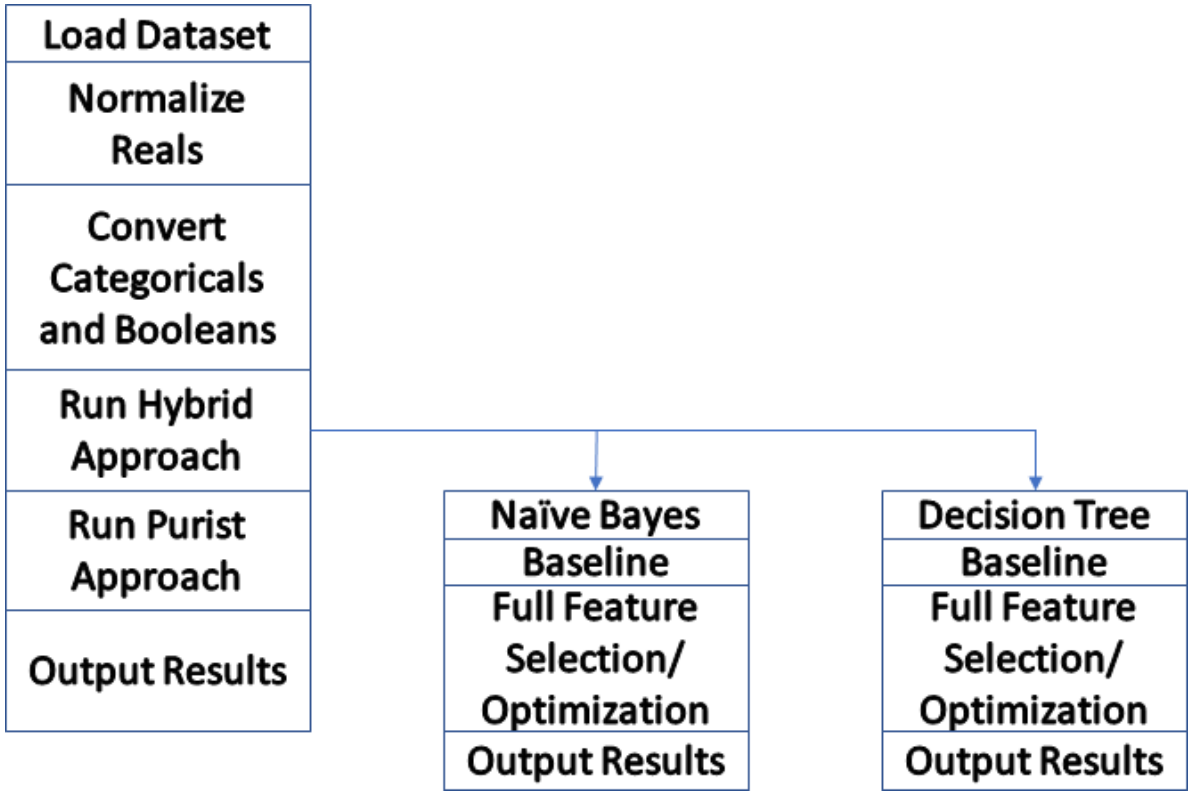


Figure: Birds-eye view of the flow of the program. Hybrid approaches are not run in parallel, because at time of coding MATLAB doesn't support multi-threading via a COM server.

For Reals

Categorical feature types:

$$
\begin{aligned}
X &= X_{\mathbb{R}} \cup X_{cat} \cup X_{bool} \\
X_{\mathbb{R}} &= \{x_1, x_2, \ldots x_n\} \\
X_{cat} &= \{x_{n+1}, x_{n+2}, \ldots x_c\} \\
X_{bool} &= \{x_{c+1}, x_{c+2}, \ldots x_b\} \\
1 &\leq i \leq c - n \\
L &= \{x_i | x_i \epsilon X_{cat}\} \\
C_\ell &= \{x, i | x \epsilon X \wedge x_{n+i} = \ell \epsilon L\} \\
R(\ell) &= \frac{\sum_{x \epsilon C_\ell} \sum_{j=1}^{n} x_j}{|C_\ell| n}
\end{aligned}
$$

Booleans

True is converted to .75, false to .25.

Outline

Hybrid Approach

- Optimizers are solutions which optimize, in this case, classifiers.
- Their purpose is to optimize which settings a classifier uses as well as selecting features.
- Each feature is represented as a bit, usually at the front of the Optimizer.
- The remainder of the genome represent parameters to the classifier
- Two classifiers, Naïve Bayes (McNB) and Classification Tree (CTree)
- Further, two versions of each: with and without feature selection (baseline)
- Fitness is defined as the average accuracy per class, or $\overline{A}$

Fitness

Consider C =

True:	ω_1	ω_2	ω_3	**Total**
Predicted ω_1	4	6	5	**15**
Predicted ω_2	2	2	9	**13**
Predicted ω_3	8	2	110	**120**
Total	**14**	**10**	**124**	**148**
TPR	0.286	0.20	0.89	.459

In this case, $A = \frac{4+2+110}{148} = 78.4\%$, but $\overline{A} = .459$, which better reflects its performance.

Outline

- Program Flow
- Optimizers
  - Naïve Bayes
  - Classification Tree
- Hunters
- Metrics
- Yeast
- Cardiotocography
- Bach's Chorales

Naïve Bayes

$$P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}$$

Where:

- $P(\omega_j|x)$ is the posterior probability that x belongs to class j
- $P(\omega_j)$ is the rate of occurrence of class j, called the prior probability.
- $p(x|\omega_j)$ is the likelihood of x belonging to ω_j
- $p(x)$ is a normalizing constant to constrain values to $\epsilon[0, 1]$

Shamelessly reproduced from Dr. Hairong Qi's ECE 471 slide

Naïve Bayes

Optimizers optimize the following:

- Features, 1 bit per feature
- Distribution, 2 bits
- Kernel, 2 bits
- Score Transform, 3 bits
- Priors, 3 bits per class

Naïve Bayes

Distribution uses 2 bits and can be one of the following:

- **Kernel** uses a smoothing function to build a distribution
- **Multinomial** represents every class as a single multinomial distribution
- **Multivariate multinomial** characterizes each feature as an independent multinomial distribution based on the unique values found in the feature.
- **Normal**

Naïve Bayes

Kernel uses 2 bits and can be one of the following:

- **Box** uses a uniform, box-like smoothing window.
- **Epanechnikov** is a very efficient, rounded kernel.
- **Gaussian** is a standard normal function but used in this case for smoothing.
- **Triangular** is another form of smoothing, with a peak of 1 at 0 and zero at -1 and 1.

Naïve Bayes

Score transform uses 3 bits and can be any of the following:

- **DoubleLogit** transforms the score to $\frac{1}{1+e^{-2x}}$
- **Invlogit** $log(\frac{x}{1-x})$
- **Logit** $\frac{1}{1+e^{-x}}$
- **None** x
- **Sign** $\frac{x}{|x|}$, or 0 when x = 0.
- **Symmetric** $2x - 1$
- **Symmetricismax** 1 if max of class, 0 otherwise
- **Symmetriclogit** $\frac{2}{1+e^{-x}} - 1$

Outline

- Program Flow
- Optimizers
  - Naïve Bayes
  - Classification Tree
- Hunters
- Metrics
- Yeast
- Cardiotocography
- Bach's Chorales

Classification Tree

- **Features**, 1 bit per feature
- **Merge Leaves**, 1 bit
- **Maximum Splits**, 6 bits
- **Min Leaf Size**, 5 bits
- **Split Criterion**, 2 bits

Classification Tree

Merge Leaves takes 1 bit and is either on or off. Merge leaves looks at leaves from a parent node and if the amount of their risk and that of their offspring is at or greater than that of a parent.

Maximum Splits defines how many splits a tree can have. The tree is built iteratively, layer by layer, splitting as needed until it hits this number. It can take on values of 3-66.

Classification Tree

Min Leaf Size This is the minimum number of samples that need to reach this node to be considered a standalone leaf. Beyond this number (specifically, at twice this number) a leaf become a parent node split into two children. Takes on values between 1 and 32.

Classification Tree

Split Criterion can take on 3 values.

- **Gini's Diversity Index** Aims for maximally diverse cuts
- **Twoing** Aims for a balanced tree
- **Deviance** Minimizes entropy

Evolver

```
AdvanceGeneration():
        EvaluateAllOptimizers(P) //Multithreaded evaluation possible
        GetMetrics()
        P.ReverseSort() //P is the population, an instance variable
        P = GenerateNextGeneration(P)
        RemoveDuplicates(P)

GenerateNextGeneration(P):
        BreedingPop := StochasticRUS(P)
        NextGen := Elitism(P)
        FillListFromBreedingPop(NextGen, BreedingPop,
                                P.Count, UniformXOver)
        MutateNonElites(NextGen)
        return NextGen
```

Evolver

```
FillListFromBreedingPop(N, B, size, Func):
E := B.Count * ElitePercent
while(N.Count < size):
        k := j := RNG.Next(0, E)
        while(j==k)
                k = RNG.Next(0, B.Count)
        for each offspring in Func(B[j], B[k])
        N.Add(offspring)

while (N.Count > size)
        N.pop()
```

Outline

Hunters

- Hunters are made up of 1 or more Chromosomes
- Chromosomes are made up of 1 or more Cells
- Cells form the bulk of the genome of the Hunter
- Fitness is a modified $\overline{A}$

Examine each component from the bottom up

Hunter Fitness

Fitness is equal to $\overline{A}$ with 2 scalars, one for complexity and one for ignoring classes:

$$F_{Hunter} = \overline{A}\left(\frac{C_{Max} - C}{C_{Max}}\right)\left(\frac{E - Z}{E}\right)$$

Outline

- Program Flow
- Optimizers
- Hunters
  - Component Structure
  - Breeding Functions
- Metrics
- Yeast
- Cardiotocography
- Bach's Chorales

Cells

Cells comprise the following:

- Function Index, $\lceil log_2(F) \rceil$ bits, where F is the number of features
- Upper Limit, 8 bits
- Lower Limit, 8 bits
- Not Flag, 1 bit
- Join Bit, 1 bit

Cells

Functions (F_i) are simply looking at features, looking at the values of the normalized dataset.
Limits are a binary number shifted to the -8th power. It allows for values between 0 and $\frac{511}{512}$. If the upper limit (L_u) is lower than the lower limit(L_ℓ), the bits are swapped.
The **Not Flag** (N) reverses the vote of the cell.
So when a cell votes, it returns $V = N \oplus (L_\ell \leq F_i \leq L_u)$
The Join Bit indicates whether to include the next cell in the voting process.

Chromosomes

Chromosomes comprise the following:

- Class Bits, $\lceil log_2(\Omega) \rceil$ bits where Ω is the number of classes
- Affinity Bits, 2
- A List of Cells

Chromosomes

Chromosomes comprise the following:

- Class Bits, $\lceil log_2(\Omega) \rceil$ bits where Ω is the number of classes
- 2 Affinity Bits, discussed in detail later
- A List of Cells

Hunters

Hunters are merely a housing for a list of Chromosomes. They don't contain any genetic information of their own. However, breeding operations and fitness is calculated at the Hunter level.

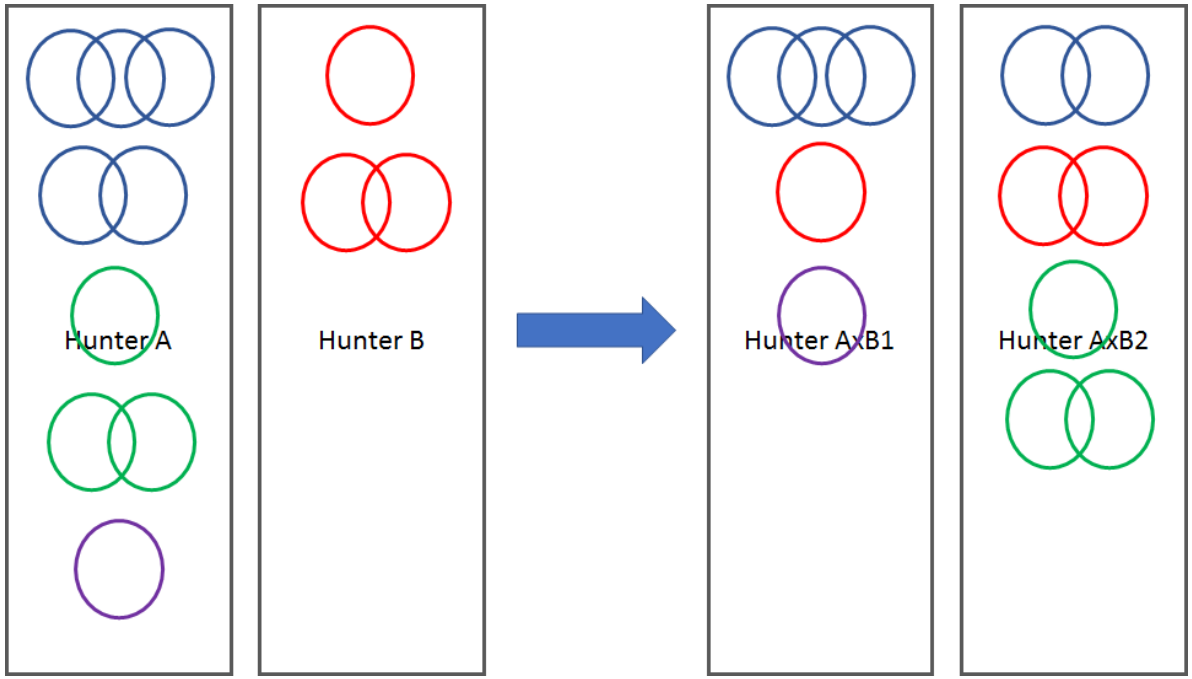
Outline

- Program Flow
- Optimizers
- Hunters
  - Component Structure
  - Breeding Functions
- Metrics
- Yeast
- Cardiotocography
- Bach's Chorales

Crossover

Crossover functions differently because Hunters are of variable length. It performs uniform crossover on the lengths which match, but then randomly assigns entire chromosomes to either offspring following the uniform crossover style.

Crossover



Merge

Merge takes 2 hunters and returns a single hunter with a combination of the material of both. The action occurs mostly at the chromosomal level.
Chromosomes have 2 affinity bits which controls the merge operation.

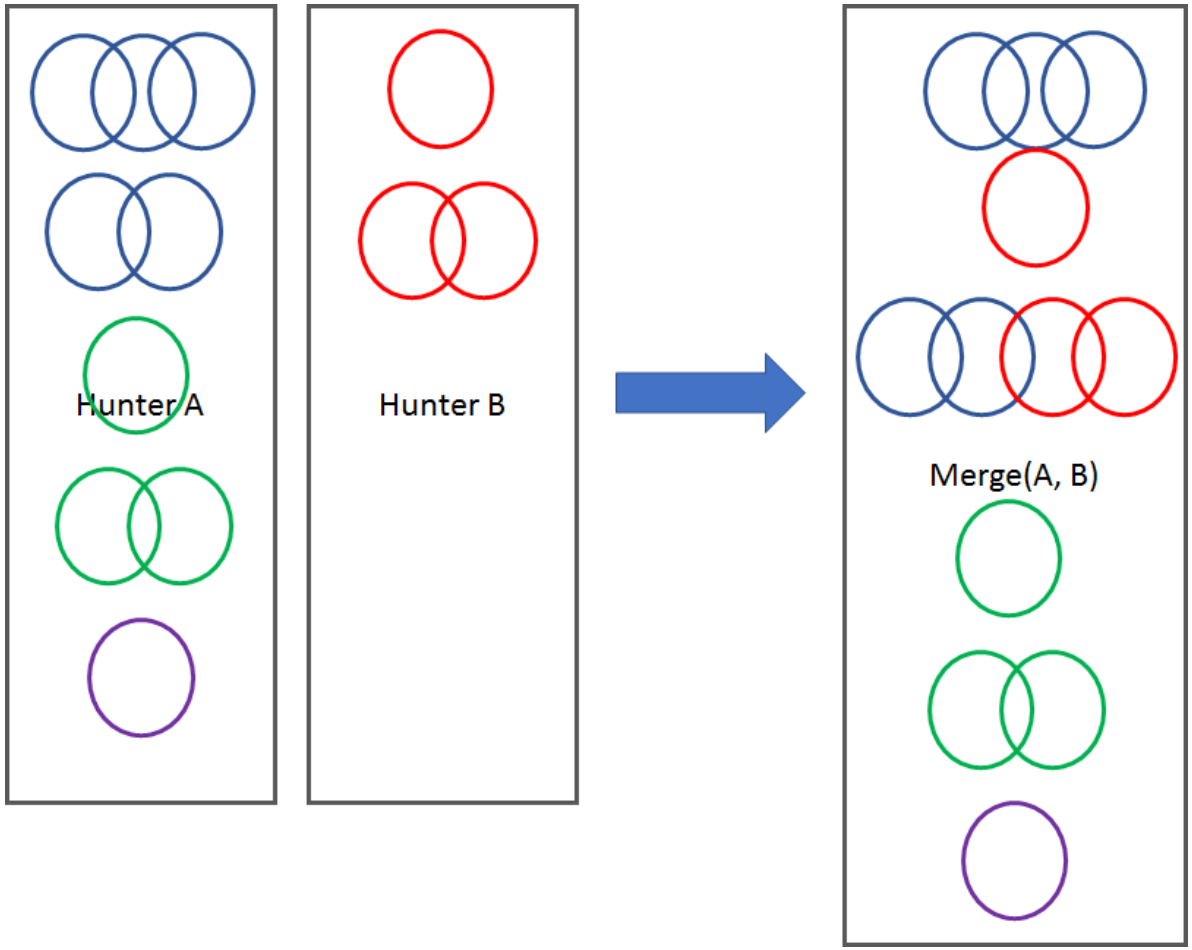
Individually, bits may be understood as follows:

		Meaning
0	0	No preference.
0	1	Prefers to be at the rear.
1	0	Prefers to be at the front.
1	1	Considers itself complete.

When two chromosomes go to merge, the results are determined as follows:

A	B	Result
00	00	
00	01	
10	00	
10	01	Laid out Horizontally with A in front
00	10	
01	00	
01	10	Laid out Horizontally with B in front
11	**	
**	11	
10	10	
01	01	Laid out Vertically

Merge



Outline

Metrics

Matthews Correlation Coefficient (MCC) and Confusion Entropy (CEN) are two promising additional methods for analyzing confusion matrices.

$$MCC = \frac{cov(X, Y)}{\sqrt{cov(X, X) \cdot cov(Y, Y)}}$$

- Falls in the range [-1,1]
- 1 indicates perfect classifier
- -1 indicates perfect anti-classifier
- 0 indicates one or more columns is equal to 0

Metrics

Consider C =

True:	ω_1	ω_2	ω_3	**Total**
Predicted ω_1	4	6	5	**15**
Predicted ω_2	2	2	9	**13**
Predicted ω_3	8	2	110	**120**
Total	**14**	**10**	**124**	**148**
TPR	0.286	0.20	0.89	.459

In this case, $A = \frac{4+2+110}{148} = 78.4\%$, but $\overline{A} = .459$, which better reflects its performance.

Outline

Outline

Outline

Outline

Outline