**A Linear Classifier for Determining the**

**Presence of Malignancy in Breast Tumors**

**Team Members:**

* Clemens Vallejo
* Zayd Hammoudeh

**Project Summary**

Use a genetic algorithm to generate a linear classifier that can determine whether a breast cancer tumor is malignant or benign.

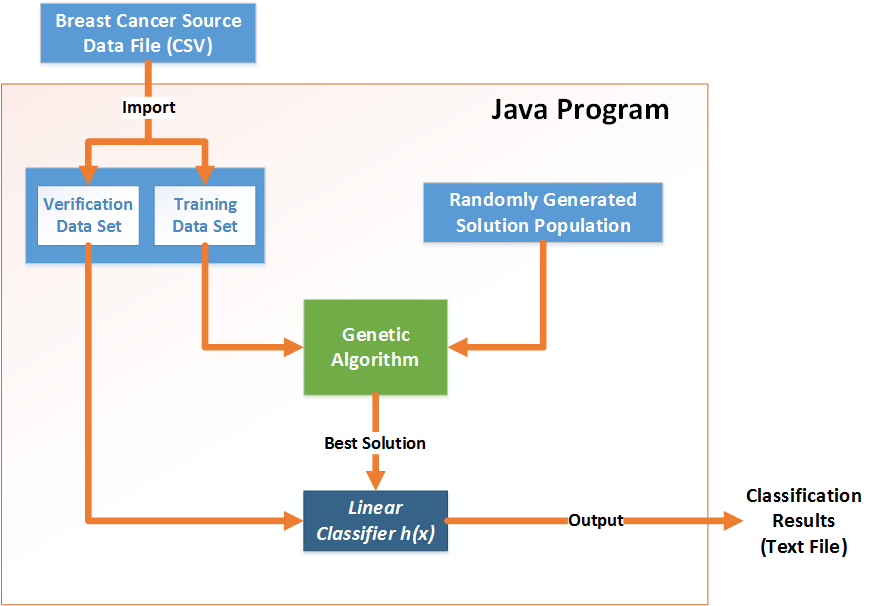
**Dataset Overview**

Our project focuses on a domain of predictive analytics known as classification. A classification algorithm uses a training data set to generate models that can classify future data members into one of a finite set of categories. In our case, we are classifying breast tumors as either malignant or benign. The source dataset is entitled the “Wisconsin Breast Cancer Database”[[1]](#footnote-1) and was created by a team from the University of Wisconsin Hospitals. The dataset consists of 699 patient tumors[[2]](#footnote-2). Each tumor has 9 distinct features each of which was assigned a value on a 1 to 10 scale; the features are listed below with brief descriptions of the biological differences between benign and malignant tumors.[[3]](#footnote-3)

1. **Clump Thickness:** Benign cells tend to be clumped in monolayers while malignant cells are usually grouped in multilayers.
2. **Uniformity of Cell Size:** Benign cells are more uniform in size while the size of malignant cells can vary significantly.
3. **Uniformity of Cell Shape:** Benign cells have smooth and round edges with a surrounding fibrous capsule that is very well-circumscribed. In contrast, the edges of the malignant cells are usually very distinct and lack this uniformity.
4. **Marginal Adhesion:** A malignant mass is mobile and not attached to surrounding tissue. Benign masses tend to stick more tightly together.
5. **Single Epithelial Cell Size:** Epithelial cells line the cavities and surfaces of structures throughout the body. In benign tumors, surrounding epithelial cells form a single layer with normal cell size; malignant tumors usually have significantly enlarged epithelial cells.
6. **Bare Nuclei:** A bare nucleus is devoid of surrounding cytoplasm (i.e. the rest of the cell). They are more typically seen in benign tumors.
7. **Bland Chromatin:** In benign tumors, the nucleus generally has a uniform texture. In contrast, cancerous cells tend to have coarser nuclei.
8. **Normal Nucleoli:** Nucleoli are small structures in the cell nucleus. In benign cells, the nucleolus is very small and barely visible (if at all). Malignant cells have more prominent nucleoli, and in some cases, they are more numerous as well.
9. **Mitoses:** It is the process in which the cell replicates and divides. Mitosis is rapid and uncontrolled in malignant cells.

**Breast Tumor Classifier Program Overview**

is a block diagram of the components of our program. The breast cancer data set, which is in Comma Separated Variable (CSV) format, is imported into our program and split into two disjoint sets (i.e. training and verification). On each run of the program, the members of the two disjoint sets are randomly selected; in all cases, the size of the training set is fixed at 200 members. By dividing the datasets randomly, we are able to gain additional insight into how population variations can impact the algorithm’s results. In our experiments to date, we have observed that the make-up of the training set does have some impact on our algorithm’s total accuracy. The effects are more severe when the training set only has a small number of malignant patients.



**Figure 1 – Block Diagram of the Components in the Breast Cancer Classifier Program**

Once the two datasets have been formed, the training data set and a set of randomly generated solutions are fed into the genetic algorithm. Random seeds tend to lead to increased solution diversity than more contrived inputs; this greater solution diversity is strongly correlated with better algorithm performance. The output of our genetic algorithm is a linear classifier (*h(x)*). The efficacy of this classifier is measured by quantifying how accurately it classified patient tumors in the verification data set.

Our genetic algorithm is implemented in the Java programming language. We selected Java because of its portability across platforms, large user base, and extensive built-in libraries. Java is primarily an object-oriented programming language; what is more, classification problems (especially genetic algorithms) are generally conducive to being programmed using an object based methodology. As such, we relied heavily on the object-oriented paradigm in our implementation. is a UML class diagram of our genetic algorithm. The five classes and their respective descriptions are listed below:

1. **BreastCancerGeneticAlgorithm** – Main program class. It is through this class that an application programmer would interact with and run the program. This class aggregates objects of the other four classes.
2. **Patient** – This eponymous class represents one patient tumor in the source dataset. Each Patient object is categorized as either malignant or benign and has values for the nine features described previously.
3. **BreastCancerDataSet** – Aggregator of objects of the Patient class. The two objects of this type are: trainingDataSet and verificationDataSet.
4. **GAChromosome** – The Genetic Algorithm (GA) chromosome encapsulates the linear classifier’s gain vector and offset scalar. Each GAChromosome object represents a possible solution; these classifiers are referred to as “chromosomes” since they undergo mutation and “sexual” reproduction where two GAChromosome objects are merged to form a successor GAChromosome object.
5. **GAChromosomePopulation** – Aggregator of objects of type GAChromosome.

**Genetic Algorithm Overview**

A genetic algorithm (GA) is a local search, learning algorithm that is modeled after the biological process of natural selection. Genetic algorithms begin with a set of randomly generated solutions to the problem. Each solution is referred to as an individual or chromosome while the set of all solutions is referred to as the population. Each solution is given a quality rating by a fitness function. Over a series of iterations (called generations), pairs of chromosomes from the previous generation (parents) are merged to form the new chromosomes that comprise the successor generation. The process where the two paternal chromosomes are merged to form the descendent chromosome is known as crossover. After crossover, the successor chromosome undergoes mutation where part of the solution is randomly changed. is pseudocode for how our algorithm implements the genetic algorithm process.

We adjusted different parameters within our genetic algorithm to determine the settings that best balanced performance with execution time. The final implementation after adjustment is described in the subsequent paragraphs.

* **Classification Strategy and Threshold:** Any patient tumor with a negative classifier function value is classified as benign while any data member whose classifier function value is positive is malignant. This approach is the simplest to implement and is the easiest for a user to understand. What is more, this approach leads to no reduction in flexibility making it an ideal implementation selection.
* **Population Size:** 1000. A large population increases solution diversity. However, as the population size increases, the incremental population diversity decreases. In our experiments, we saw little to no improved performance when the population size was larger than 1000 members.

**for** *restart\_number* = 1 **to** *NUMBER\_RANDOM\_RESTARTS* **do**

*population* ← generate *N* random chromosomes

**for** *generation\_number* = 1 **to** *MAX\_NUMBER\_GENERATIONS* **do**

evaluate all chromosomes using the fitness functions

copy *M* best chromosomes to *new\_population*

**while** *size*(*new\_population*) ≤ *MAX\_POPULATION\_SIZE* **do**

*parent1* ← select chromosome

*parent2* ← select chromosome

*child* ← *crossover*(*parent1* , *parent2* )

potentially randomly mutate *child*

add *child* to *new\_population*

**end while**

*population* ← *new\_population*

**end for**

**if**(*fitness*(*best\_solution*) < *fitness*(*population*.best\_solution))

*best\_solution* = *population*.best\_solution

**end for**

**return** *best\_solution*

Figure – Breast Cancer Genetic Algorithm Pseudocode

* **Number of Generations:** 1000. We saw solution convergence below 1000 generations so increasing the number of generations did not lead to improved performance.
* **Random Restarts:** 5 – By allowing multiple random restarts, the algorithm’s solution diversity is increased which in turn increases the likelihood of bypassing local minima. We generally saw solution convergence between the second and fourth random restart.
* **Chromosome (Solution) Structure:** The data set has 9 features, and there is an additional term in the linear classifier function for the offset. In our algorithm, each weight is a Java integer (i.e. 32-bits, two’s complement). Hence, given the 9 features and the offset term, a chromosome is 320 (i.e. 32 \* 10) bits long.
* **Crossover Operator:** In our implementation, the number of crossover points is an adjustable variable. Excluding the case of one-point crossover, we did not observe a strong correlation between the number of crossover points and the classification accuracy. As such, we selected three-point crossover since it provides an adequate balance between classification accuracy and execution time.
* **Mutation Frequency:** A higher mutation frequency is correlated with increased solution diversity. However, if the mutation frequency is too high, it can have a deleterious effect because it can corrupt otherwise good solutions. The bit mutation frequency is set to 1% in our solution, which while high showed no deleterious effects on the algorithm’s accuracy.
* **Fitness Function** – The fitness function, *f*, measures the quality of any solution. Eq. is our current fitness function.

|  |  |
| --- | --- |
|  | ( 1 ) |

where *M* is the size of the population, *i* represents the *i*th data member in the dataset, and  is the value linear classifier function for patient vector, . is a normalizing scalar defined by the relation in eq. ; this normalizing scalar is used to reward correct classification of benign tumors, which will have negative scores. Given the preliminary results that are described in the next section, we may adjust our fitness function to improve its performance on malignant tumors.



|  |  |
| --- | --- |
|  | ( 2 ) |

* **Reproduction Selection Algorithms:** Tournament selection in a genetic algorithm involves randomly picking *n* chromosomes from the population; from these *n* possible solutions, the chromosome with the highest fitness is selected to be a parent of a successor chromosome (solution). Tournament selection has low computational overhead and prevents the algorithm converging too quickly (assuming *n* is not too large relative to the population size). For these reasons, we used this approach as the natural selection paradigm in our algorithm.

**Genetic Algorithm Preliminary Results**

The output of the genetic algorithm is a single chromosome (i.e. linear classifier). This linear classifier is used to classify all patient tumors in the verification data set as either malignant or benign. There are two primary metrics we used to quantify the quality of this classification. They are:

1. **Total Accuracy** – This is the most straightforward quality measure and is simply the ratio of the number of correct classifications to the total size of the verification data set.
2. **Malignancy Classification Accuracy (MCA)** – The most serious type of classification error is a false negative, where a tumor is classified as benign when it is really malignant. This metric quantities the accuracy of the algorithm with respect to this type of error.

**Error! Reference source not found.** is an example output from our program. For each random restart, the total accuracy on the training set is provided. After the five random restarts, the total accuracy and malignancy classification accuracy (MCA) is printed along with the weights of our linear classifier. Table 1 shows the performance of our algorithm over multiple iterations (where an iteration is defined as the creation of a new, randomly selected training data set). In almost all cases, our total accuracy was greater than 90%. In most cases, the MCA was less than the total accuracy. We hypothesize that this is due to malignant tumors being under-represented in the training set. To improve the MCA, we are considering three modifications to our algorithm:

1. **Increase the Size of the Training Set:** The training data set is currently about half the size of the verification set, and some adjustments to this ratio may improve the accuracy our system. However, our initial experiments did not indicate that this approach would lead to statistically significant improvements.
2. **Maintain Proportionality of Malignant Tumors between the Training and Verification Data Sets:** Currently, patients are randomly selected for inclusion in the training data set. As such, malignant tumors may be under or over represented in that data set. We are considering adjusting the population selection algorithm to require that the percentage of cancerous and non-cancerous patients be the same in both the training and verification data sets.
3. **False Negative Penalty**: Since a false negative is far more serious than a false positive, we are considering assigning different weights to the two types of errors. This approach would increase the malignancy classification accuracy (MCA); however, it may also decrease the algorithm’s total accuracy.

|  |  |  |
| --- | --- | --- |
| Iteration # | Total Accuracy (%) | MCA (%) |
| 1 | 91.3% | 91.8% |
| 2 | 89.9% | 86.4% |
| 3 | 90.1% | 86.2% |
| 4 | 91.7% | 85.0% |
| 5 | 94.0% | 95.1% |
| 6 | 90.7% | 85.3% |
| 7 | 91.7% | 94.8% |
| 8 | 91.9% | 89.8% |
| 9 | 90.1% | 88.4% |
| 10 | 91.3% | 85.9% |
| Average | 91.3% | 88.9% |

Table – Breast Cancer Classifier Performance across Multiple Iterations

**Conclusions**

Even without further modification, the classification accuracy of our algorithm is high (~90%). Since the algorithm is fully implemented and working, future efforts will be focused on optimizing the algorithm’s classification accuracy, in particular malignancy classification.

Appendix A – Genetic Algorithm Sample Output

Below is a sample output of the genetic algorithm.

After 1 run, the percent correct on the training set is: 97.00

After 2 run, the percent correct on the training set is: 97.00

After 3 run, the percent correct on the training set is: 97.00

After 4 run, the percent correct on the training set is: 97.00

After 5 run, the percent correct on the training set is: 97.00

On the training set, the score for the best solution is: 194

The maximum possible score is: 200

The percent correct is: 97.00

The linear function weights are:

Clump Thickness Weight: -73149407

Cell Size Uniformity Weight: 806500191

Cell Shape Uniformity Weight: 383976807

Marginal Adhesion Weight: 7331932

Single Epithelial Cell Size Weight: -714736227

Bare Nucleoli Weight: 476423725

Bland Chromatin Weight: -100514872

Normal Nucleoli Weight: 48835613

Mitoses Weight: -52238695

Offset Weight: -2094917374

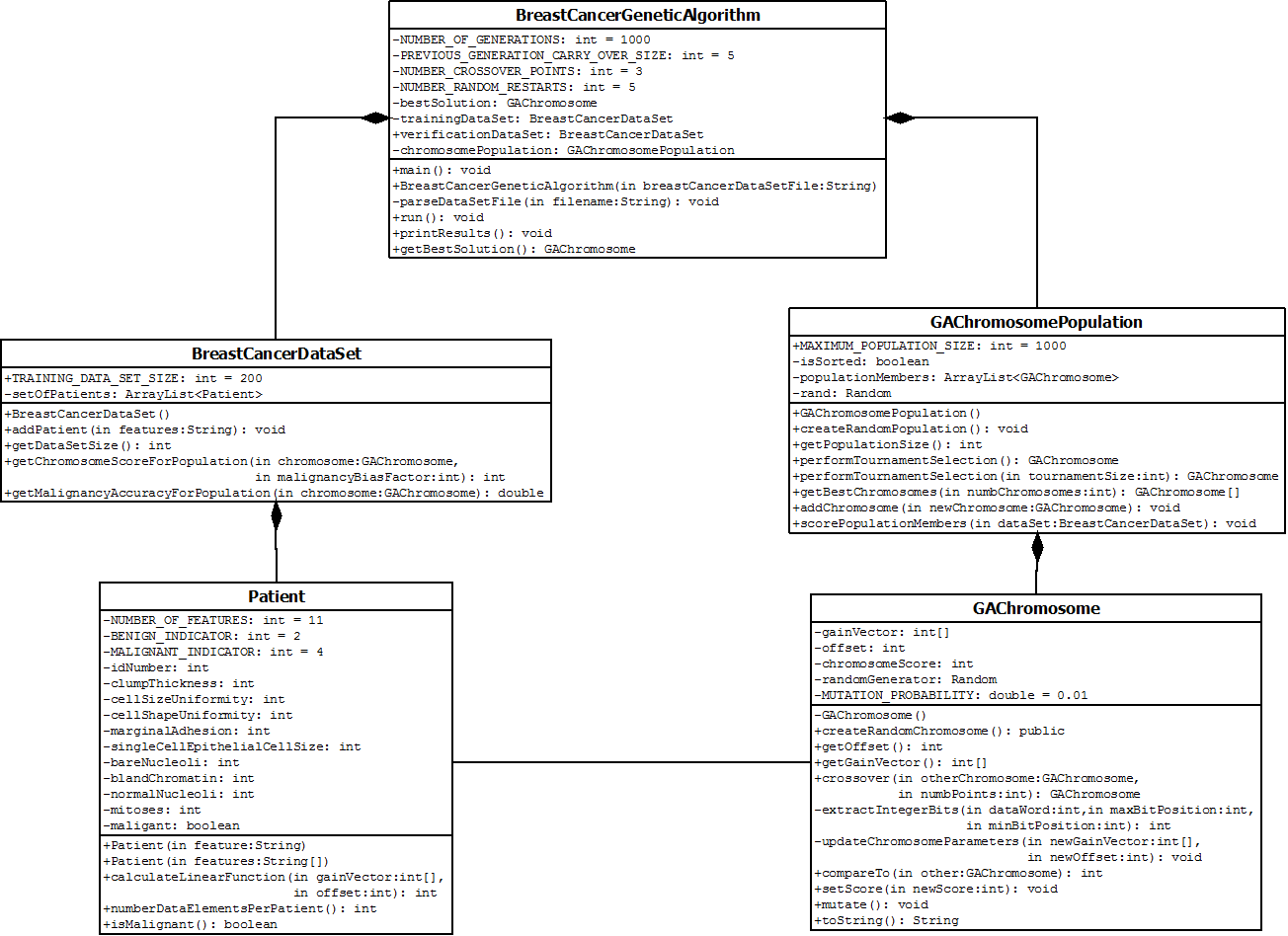
On the verification set, the score for the best solution is: 456

The maximum possible score is: 483

The percent correct is: 94.41

The percentage of malignant tumors correctly categorized is: 92.66%.

Appendix B – UML Class Diagram for the Breast Cancer Classifier Genetic Algorithm



1. The dataset is available in the University of California, Irvine’s Machine Learning Repository. A link to the dataset is: <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original)>. [↑](#footnote-ref-1)
2. 16 of the 699 tumors have incomplete data so only 683 instances are used in our analysis. [↑](#footnote-ref-2)
3. **Reference:** Salama et. al. “Breast Cancer Diagnosis on Three Different Datasets Using Multi-Classifiers.” *International Journal of Computer and Information Technology*. September 2012. [↑](#footnote-ref-3)