Metaheuristics in Synthetic Biology

Report by Anthony Hamilton Fernando

1.Introduction

In the Softwarepraktikum "Metaheuristics in Synthetic Biology" we worked with so-called cell classification. These can be used to predict whether a cell is cancerous or healthy. We looked at binarized expression level data of different miRNA species in the cell. In our case it was liver cancer cell data. To find the classifier, we use a so-called genetic algorithm. These are population-specific metaheuristics based on Darwin's evolution theory and are used for optimization and search problems. With the help of a self-implemented genetic algorithm, we try to find the perfect cell classifier from the given data. This consists of different miRNAs and the correct prediction is done by boolean logic. The procedure is carried out in such a way that we have a training data and with the genetic algorithm we find our best classifiers. At the end we take the perfect classifier with a maximum of 10 classifiers and test it on the test data to see how good it is. With this classifier you can later create biological synthetic logic circuits. They can classify whether the cell is cancerous (1) or healthy (0) by the expression level of miRNas in the cell. In case of cancer, apoptosis is then triggered.

2.Description of the algorithm

2.1 Description of the general design

The developed algorithm consists of 5 steps: (1 - Data Cleaning) the training data will be cleaned, that means to remove all columns or miRNAs consisting of only zeros or only ones from the training data. (2 - Initial Population) Based on the variable “classifier size “ (varies between 5 and 10) and “population size” (varies between 50 and 300) an initial population was generated randomly. In other words, 5 to 10 miRNAs are randomly selected. This is an individual consisting of zeros and ones. (3 - Evaluation) Each individual of this population is then evaluated by the F1 score. The higher the score is, the better the individual is. At the end of this step, the individuals are sorted in descending order based on their score. (4 - Selection) The first (best) half of the population is selected. (5 - Crossover). For every two following individuals Uniform Crossover was applied. For each application, two new individuals are generated and inserted into the selected population (see step 4). This results in a new population that has the same size as the initial population. In case the size of the new population is larger than the initial population, the first X individuals of the new population are selected, where X is the size of the initial population. (6- Mutation) For each individual swap mutation is used, where 2 random positions (miRNAs) of that individual are swapped.

Steps 3- 6 are repeated until all iterations are done. This algorithm is repeated for every 100 randomly generated combinations of the following parameters: NUMBER_OF_ITERATION, POPULATION_SIZE, MUTATION_PROB, TAUSCH, CLASSIFIER_SIZE. At the end, after all combinations have been performed, the parameters with the best score of all combinations are selected. This parameter was then applied 10 times to the training data and the best individual was selected based on the F1 score. This individual, called Test classifier, is tested on the test data. By majority vote it is predicted for each individual in the test data whether

they are healthy or have cancer. The prediction is compared with the annotation to see how good the classifier is.

2.2 Pseudocode

Algorithm 1: Genetic Algorithm

```
Data: Binarized DataSet
Result: Best classifier contains a minimum of 5 to 10 miRNAs
Def main_train(train_dataset, NUMBER_OF_ITERATION,
 POPULATION_SIZE, MUTATION_PROB, TAUSCH,
 CLASSIFIER_SIZE):
    train_data = clean_data(train dataset)
    train_data = Select_randomly_miRNAs(train_data,
     CLASSIFIER_SIZE)
    random_data = initialization(train_data, size,
     POPULATION_SIZE, CLASSIFIER_SIZE)
    for i := 1 to NUMBER_OF_ITERATION do
        res = Evaluation(train_data, random_data,
         NUMBER_OF_ITERATION, TAUSCH, MUTATION_PROB)
        selectiongroup = Selection(random_data)
         //select half of the population
        positions_list = Create_randomly_vector(selectiongroup, row,
         TAUSCH)
        df_crossover = Crossover(selectiongroup, positions_list)
        df_mutation = Mutation(df_crossover, MUTATION_PROB)
    end
    return result
```

1. train dataset= binarized dataset
2. NUMBER_OF_ITERATION= numbers of iteration in genetic algorithm
3. POPULATION_SIZE= # individual
4. MUTATION_PROB= mutation probability
5. TAUSCH= # of miRNAs to be swapped
6. CLASSIFIER_SIZE= # of miRNA from each individual
7. random_data= random generated population
8. res= evaluated population
9. selectiongroup= selected individuals
10. position_list= position from individual to be swapped
11. df_crossover= crossover population
12. df_mutation= mutation population

Note: This pseudocode calculates only one parameters combination. To find the best classifier, all parameters combinations must be run through first. From these, the best parameters combination is chosen that has achieved the best score. With this parameter combination, this is then run through 10 times to get the best classifier

2.3 Operators and Parameters

Population

The generation of the population will be random. The size of the population ranges from min 50 to max 300 individuals in steps of 50. Each individual has a certain number of miRNAs. These miRNAs are randomly annotated with 0 or 1. Thereby 0 should be downregulated and 1 upregulated. Each individual has either a minimum of 5 randomly selected miRNAs up to a maximum of 10 miRNAs. To get a general solution it is always best to work randomly, i.e. different numbers of miRNAs which are randomly up- or down-regulated. In order to evaluate the results on different classifier size, the interval of 5-10 is chosen.

1. [0,1,0,1,1]
2. [0,1,0,1,0,0,0] <- an example of a population
3. [0,1,0,1,1,0,1,1]

Evaluation function

After an initial population was randomly generated, the fitness values between the random individuals and rows in training data were evaluated. This was done by calculating the F1 score. For each individual in the random population, the F1 score is calculated with each line in the training data. F1 Score formula is defined as 2 * TP / (2 * TP + FP + FN), where TP is True Positive, FP is False Positive, and FN is False Negative. To calculate the TP, FP FN between a random individual and one row of the training data, the following applies:

(Case 1 - Annotation is 0): The number of both similar and unsimilar values in each position(miRNA) in both vectors is counted. These two values are saved in Match (=) and unmatch (≠) variables respectively. If unmatch is bigger or equal than match then TN is equal to 1, otherwise FP is equal to 1.

(Case 2 - Annotation is 1): Similar to the previous case but here if match is bigger or equal than unmatch then TP is equal to 1, otherwise FN is equal to 1.

To compute the F1 score for a particular individual, we need first to sum up all TP, TN, FP, FN values. That is the F1 score can be computed using the formula defined above and by using the summed values of TP, TN, FP, FN. F1 Score is a good Score to use if it is important to balance between the precision and recall, that are calculated using TP,FP,TN and FN.

Individual: [1,1,1,1,0]

Trainings data:

0[0,1,0,1,0] 3= 2≠ FN

1[1,1,0,1,1] 3= 2≠ TP

0[1,0,0,0,1] 1= 4≠ TN

1[0,0,1,0,0] 4≠ 1= FP

At the end of this step, the individuals are sorted in descending order based on their score.

Selection

As the created population from the last step is sorted, the (first) half elements of this population having the best scores are selected. These new selected elements are now called crossover population. This population is the half size of the random generated population from the last step. This means that if the initial population consists of 100 individuals, the crossover population consists of 50 individuals which are determined by the fitness. It is assumed that the crossover population will minimally improve by choosing the individuals with the best F1 scores.

Crossover

The crossover population from the last step contains individuals called parents. Uniform crossover is applied to each subsequent parent. Uniform crossover generates a random position of the parent pairs. The number of random positions is decided by the "TAUSCH" variable. The elements at the randomly generated positions of both parents are swapped with each other. This creates two new individuals called children. In other words, for every two parents two new children are created. The TAUSCH variable varies randomly between 3 and 5.

As an example: Let “TAUSCH” be 3 and let 0, 1 and 5 be the randomly generate positions.

Parent_1 [0,0,1,1,0,0] -> [1,1,1,1,0,1] Child_1

Parent_2[1,1,1,0,1,1] -> [0,0,1,1,0,0] Child_2

In the end, a mutation population is created, where the first half consists of parents and the second half of children. The reason why uniform crossover is used is that the random number of items exchanged creates new individuals, which leads to a higher variation.

Mutation

A swap mutation is used for each element of the mutation population. For each individual two randomly positions are created and the elements (miRNAs) at these positions will be swapped with each other. However, the swapping can only be applied if a randomly generated probability (a float number between 0 and 1) is bigger than the defined “MUTATION_PROB” variable. Again, randomness is chosen for the creation of a new population.

As an example:

Let “MUTATION_PROB” be 0.5, random probability be 0.6, and let 2 and 5 be the randomly generated swap positions. Because random probability is bigger than the “MUTATION_PROB” miRNAs at positions 2 and 5 will be swapped.

[0,0,1,1,0,0] -> [0,0,0,1,0,1]

Parameter tuning

The following ranges are defined for the previous mentioned parameters:

1. NUMBER_OF_ITERATION : 50-100 in step of 25
2. POPULATION_SIZE: 50-30 in step of 50
3. MUTATION_PROB: 0.1 -1.0 in step of 0.1
4. TAUSCH: 3-5 in step of 1
5. CLASSIFIER_SIZE: 5-10 in step of 1

3.Results

Liver_train_1:

Table X1: The best F1 scores of each Parameter combination

number of Iteration	Population size	mutation_prob	Tausch	classifier size	F1	AVG F1
75	100	0.2	5	8	0.967741	0.92672
75	300	0.2	5	8	0.96	0.90845
100	150	0.9	4	6	0.959349	0.88110
100	150	0.30000000000	3	6	0.959349	0.92286
100	200	0.4	3	6	0.959349	0.91051
50	250	0.5	3	8	0.958677	0.87748
50	250	0.5	4	8	0.958677	0.87516
25	100	0.70000000000	3	8	0.958677	0.81112
50	200	0.9	4	6	0.958677	0.88245
75	250	0.6	4	10	0.951612	0.88491
25	200	0.70000000000	4	10	0.951612	0.80294
50	300	0.1	3	10	0.951612	0.88508
50	200	0.8	3	10	0.951612	0.85262
100	150	0.8	5	10	0.951612	0.88925
50	150	0.2	4	10	0.951612	0.87747
75	200	0.4	3	10	0.951612	0.89634
50	250	0.30000000000	5	10	0.951612	0.87700
50	150	0.2	4	10	0.951612	0.88523
100	100	0.70000000000	3	10	0.951612	0.89730
100	200	0.5	4	10	0.951612	0.90142
100	250	1.0	3	10	0.951612	0.88591
50	300	1.0	5	10	0.951612	0.84402
50	100	1.0	3	10	0.951612	0.85816
25	100	0.30000000000	4	10	0.951612	0.82278
50	200	0.5	5	8	0.950819	0.87779
25	200	0.1	3	10	0.950819	0.84213
25	150	0.5	5	8	0.950819	0.82900
50	100	0.9	3	8	0.950819	0.86152
25	50	0.9	5	8	0.95	0.79067
25	50	0.2	3	8	0.944	0.81002
50	200	0.6	5	8	0.944	0.85605
100	150	0.5	3	8	0.942148	0.88291
50	200	0.30000000000	3	8	0.942148	0.88428
50	100	0.4	5	6	0.942148	0.88116
100	100	0.8	4	8	0.941176	0.88627
50	50	1.0	3	6	0.932203	0.81642
75	100	1.0	3	8	0.932203	0.87347
50	150	0.30000000000	5	8	0.932203	0.85721
50	50	0.6	5	6	0.931034	0.84506
100	250	0.30000000000	5	6	0.931034	0.89389
100	300	0.1	3	9	0.926829	0.88856
25	150	0.70000000000	5	5	0.925619	0.62933
50	250	0.5	4	6	0.925619	0.85600
50	200	0.9	3	6	0.925619	0.85151
25	150	1.0	5	6	0.925619	0.79229
25	100	0.2	3	6	0.924369	0.83084
50	200	0.5	5	6	0.923076	0.85079
25	100	0.8	5	6	0.923076	0.78825
75	100	0.9	3	6	0.923076	0.84733
50	100	0.5	3	9	0.915254	0.82188

Table X2: Best Parameter training 10 times

classifiers	F1	AVG F1	miRNAs
0.0 1.0 1.0 0.0 1.0 0.0 1.0 0.0	0.958677685	0.910540291	hsa-miR-100 hsa-miR-224 hsa-miR-93 hsa-miR-422a hsa-miR-106b hsa-miR-106a hsa-miR-221 hsa-miR-99a
0.0 1.0 1.0 1.0 1.0 0.0 0.0 1.0	0.950819672	0.909124271	hsa-miR-99a hsa-miR-18a hsa-miR-221 hsa-miR-106b hsa-miR-222 hsa-miR-106a hsa-miR-100 hsa-miR-93
1.0 1.0 1.0 0.0 1.0 0.0 0.0 1.0	0.950819672	0.908222606	hsa-miR-18a hsa-miR-224 hsa-miR-222 hsa-miR-106a hsa-miR-221 hsa-miR-99a hsa-miR-100 hsa-miR-106b
1.0 1.0 0.0 1.0 0.0 1.0 1.0 0.0	0.95	0.909209756	hsa-miR-221 hsa-miR-224 hsa-miR-100 hsa-miR-106b hsa-miR-422a hsa-miR-93 hsa-miR-18a hsa-miR-99a
1.0 0.0 1.0 1.0 1.0 0.0 0.0 1.0	0.943089430	0.894625261	hsa-miR-106b hsa-miR-106a hsa-miR-221 hsa-miR-422a hsa-miR-222 hsa-miR-99a hsa-miR-100 hsa-miR-18a
0.0 0.0 nan 1.0 1.0 1.0 0.0 1.0	0.942148760	0.896204899	hsa-miR-106a hsa-miR-422a hsa-miR-100 hsa-miR-18a hsa-miR-93 hsa-miR-222 hsa-miR-99a hsa-miR-106b
0.0 0.0 nan 1.0 0.0 1.0 0.0 1.0	0.941176470	0.895294970	hsa-miR-422a hsa-miR-99a hsa-miR-18a hsa-miR-224 hsa-miR-106a hsa-miR-106b hsa-miR-100 hsa-miR-221
1.0 0.0 1.0 0.0 1.0 1.0 1.0 0.0	0.941176470	0.906065661	hsa-miR-221 hsa-miR-100 hsa-miR-18a hsa-miR-422a hsa-miR-106b hsa-miR-224 hsa-miR-93 hsa-miR-222
1.0 0.0 0.0 0.0 1.0 1.0 1.0 1.0	0.940170940	0.900724540	hsa-miR-222 hsa-miR-106a hsa-miR-422a hsa-miR-100 hsa-miR-93 hsa-miR-221 hsa-miR-18a hsa-miR-224
1.0 0.0 0.0 1.0 1.0 0.0 1.0 1.0	0.933333333	0.886037662	hsa-miR-221 hsa-miR-106a hsa-miR-100 hsa-miR-106b hsa-miR-93 hsa-miR-422a hsa-miR-222 hsa-miR-18a

Note: "nan" means that this miRNAs is not used.

Table X3: Best Classifier testing and predicting on liver_test_1

ID	Annots	hsa-miR-100	hsa-miR-224	hsa-miR-93	hsa-miR-422a	hsa-miR-106b	hsa-miR-106a	hsa-miR-221	hsa-miR-99a	prediction_int	prediction_str
3	0	1	0	0	0	0	0	0	0	0	healthy
5	0	1	0	0	1	1	0	0	1	0	healthy
9	0	1	0	0	1	0	0	0	1	0	healthy
16	0	1	0	0	1	0	0	0	0	0	healthy
29	0	1	0	0	0	0	0	0	1	0	healthy
31	0	1	0	0	1	0	0	1	1	0	healthy
36	0	1	0	0	1	1	0	0	1	0	healthy
41	0	0	0	0	0	0	0	0	0	0	healthy
49	0	1	0	0	1	0	0	0	1	0	healthy
55	0	1	0	0	1	0	0	1	1	0	healthy
57	0	1	0	0	1	0	0	0	1	0	healthy
60	0	1	0	0	0	1	0	0	0	0	healthy
65	0	0	0	0	1	1	0	0	1	0	healthy
70	0	1	1	1	1	1	0	1	1	1	cancer
72	0	1	0	0	1	0	0	1	1	0	healthy
74	0	1	1	1	1	1	0	1	1	1	cancer
81	1	1	0	1	0	1	0	1	0	1	cancer
83	1	0	1	1	1	1	1	1	0	1	cancer
87	1	0	1	0	0	1	1	1	0	1	cancer
94	1	1	1	1	0	1	1	1	0	1	cancer
107	1	0	1	0	0	0	0	0	0	1	cancer
109	1	1	0	0	1	0	0	0	1	0	healthy
114	1	1	0	1	1	1	0	1	0	1	cancer
119	1	0	1	0	0	0	0	1	0	1	cancer
127	1	0	0	1	0	1	1	0	0	1	cancer
133	1	1	0	0	1	0	0	1	1	0	healthy
135	1	1	1	1	0	1	0	1	0	1	cancer
138	1	0	1	1	0	1	1	1	0	1	cancer
143	1	0	1	1	0	1	1	0	0	1	cancer
148	1	1	0	1	0	1	1	1	1	0	healthy
150	1	1	0	1	0	1	0	1	0	1	cancer
152	1	0	0	1	0	1	0	1	0	1	cancer

Liver_train_2:

Table Y1: The best F1 scores of each Parameter combination

number of Iteration	Population size	mutation_prob	Tausch	classifier size	F1	AVG F1
50	150	0.1	3	8	1.0	0.9459153
100	50	0.4	4	10	1.0	0.9451605
25	100	0.8	3	10	0.993377	0.8240329
50	200	0.1	3	10	0.993377	0.9308899
50	50	0.70000000000	4	10	0.993377	0.8924036
100	250	0.30000000000	4	8	0.993377	0.9581283
100	250	0.2	5	10	0.993377	0.9518059
25	50	0.8	3	9	0.986666	0.7695887
50	150	0.5	5	8	0.986666	0.8949717
75	200	0.30000000000	3	10	0.986666	0.9311292
75	300	0.70000000000	4	8	0.986666	0.9158758
75	150	0.9	4	8	0.986666	0.9238097
25	50	0.9	3	10	0.986666	0.8233554
25	100	0.8	3	8	0.986666	0.8542245
100	300	1.0	4	7	0.986666	0.8269530
75	100	0.2	5	10	0.986666	0.9396861
75	300	0.1	4	8	0.986666	0.9537143
25	150	0.8	4	7	0.980132	0.7694824
75	50	1.0	3	10	0.980132	0.8979934
75	50	1.0	3	7	0.979865	0.8011606
25	100	0.4	4	10	0.979865	0.8335095
75	300	0.2	3	9	0.979865	0.9258634
50	300	0.4	4	8	0.979865	0.9083674
25	200	1.0	3	10	0.979865	0.7982882
100	50	0.4	5	6	0.973333	0.8885262
100	100	0.4	3	8	0.973333	0.9233626
25	250	0.1	4	10	0.973333	0.8383036
50	150	1.0	3	9	0.972972	0.8431520
75	100	0.9	5	10	0.972972	0.8821552
75	100	0.8	3	8	0.972972	0.9012301
25	50	0.8	5	10	0.972972	0.8037307
100	100	0.9	3	8	0.972972	0.8936630
75	50	1.0	3	6	0.972972	0.8188872
75	100	0.2	4	10	0.972972	0.9228545
100	150	0.9	3	6	0.972972	0.8825743
50	50	0.6	4	7	0.972972	0.8760557
50	300	0.70000000000	5	6	0.966887	0.8601280
50	250	0.1	3	9	0.966442	0.8942253
50	150	0.2	5	10	0.965986	0.9019869
25	250	0.70000000000	4	6	0.965986	0.8480410
25	100	0.4	5	6	0.965986	0.8572235
75	250	0.5	4	8	0.965986	0.8948884
50	200	0.2	3	9	0.965986	0.8902799
50	300	0.2	4	8	0.965986	0.9005054
100	300	0.8	4	7	0.965986	0.8739381
75	250	1.0	5	6	0.965986	0.8863592
75	100	0.5	3	6	0.965986	0.9042160
100	50	0.5	5	7	0.960526	0.9311738
50	200	0.4	3	9	0.958904	0.8669397
75	150	1.0	5	9	0.951724	0.8464180

Table Y2: Best Classifier testing on liver_test_2

classifiers	F1	AVG F1	miRNAs
0.0 0.0 1.0 1.0 1.0 0.0 1.0 0.0	1.0	0.939434	hsa-mir-199a* hsa-mir-196b hsa-mir-30d hsa-mir-188 hsa-mir-301 hsa-mir-199b hsa-mir-151 hsa-mir-26a
0.0 0.0 1.0 1.0 1.0 1.0 0.0 1.0	0.980132	0.924933	hsa-mir-107 hsa-mir-200b hsa-mir-143 hsa-mir-301 hsa-mir-339 hsa-mir-224 hsa-mir-199a* hsa-mir-183
0.0 0.0 0.0 0.0 1.0 1.0 0.0 1.0	0.972972	0.911016	hsa-mir-422b hsa-mir-143 hsa-mir-125a hsa-mir-107 hsa-mir-21 hsa-mir-151 hsa-mir-200b hsa-mir-196b
1.0 0.0 0.0 1.0 nan 0.0 0.0 0.0	0.966442	0.913212	hsa-mir-100 hsa-mir-214 hsa-mir-125b hsa-mir-18a hsa-mir-302c* hsa-mir-424 hsa-mir-223 hsa-mir-222
0.0 1.0 1.0 1.0 0.0 1.0 0.0 1.0	0.965986	0.906485	hsa-mir-450 hsa-mir-221 hsa-mir-128b hsa-mir-222 hsa-mir-422b hsa-mir-452 hsa-mir-130a hsa-mir-103
0.0 0.0 0.0 0.0 0.0 1.0 1.0 1.0	0.958904	0.905892	hsa-mir-200b hsa-mir-146a hsa-mir-223 hsa-mir-7 hsa-mir-200a hsa-mir-331 hsa-mir-302c* hsa-mir-15b
0.0 1.0 0.0 1.0 0.0 0.0 0.0 1.0	0.958904	0.899234	hsa-mir-199b hsa-mir-195 hsa-mir-378 hsa-mir-222 hsa-let-7c hsa-mir-302c* hsa-mir-199a hsa-mir-191
1.0 0.0 1.0 1.0 1.0 1.0 1.0 1.0	0.951724	0.895114	hsa-mir-30d hsa-mir-99a hsa-mir-93 hsa-mir-221 hsa-mir-222 hsa-mir-103 hsa-mir-32 hsa-mir-196b
0.0 0.0 0.0 0.0 0.0 0.0 1.0 1.0	0.937062	0.889476	hsa-mir-214 hsa-mir-200b hsa-let-7c hsa-mir-130a hsa-mir-199b hsa-mir-195 hsa-let-7b hsa-mir-130b
1.0 0.0 0.0 1.0 1.0 1.0 0.0 1.0	0.932432	0.875288	hsa-mir-199b hsa-mir-130a hsa-mir-17-5p hsa-mir-7 hsa-mir-331 hsa-mir-183 hsa-mir-200a hsa-mir-151

Table Y3: Best Classifier testing and predicting on liver_test_1

ID	Annots	hsa-mir-199a	hsa-mir-196b	hsa-mir-30d	hsa-mir-188	hsa-mir-301	hsa-mir-199b	hsa-mir-151	hsa-mir-26a	prediction_int	prediction_str
7	0	1	0	1	0	0	1	0	1	0	healthy
9	0	1	0	0	0	0	1	0	1	0	healthy
21	0	1	0	0	0	0	1	0	1	0	healthy
22	0	1	0	0	0	1	1	0	1	0	healthy
25	0	1	0	0	0	0	1	0	0	0	healthy
30	0	1	0	1	0	0	1	0	1	0	healthy
32	0	1	0	1	0	0	1	0	1	0	healthy
36	0	1	0	0	0	0	1	0	1	0	healthy
40	0	1	0	0	0	0	1	0	0	0	healthy
47	0	1	0	0	0	0	1	0	0	0	healthy
52	0	1	0	0	0	0	1	0	1	0	healthy
58	0	1	0	0	0	0	1	0	1	0	healthy
61	0	1	0	0	0	0	1	0	1	0	healthy
66	0	0	0	0	0	0	1	0	1	0	healthy
68	0	1	0	1	0	0	1	1	1	0	healthy
70	0	1	0	1	0	0	0	0	1	0	healthy
73	0	1	0	1	0	0	1	0	1	0	healthy
79	0	1	1	0	0	1	1	0	1	0	healthy
87	0	1	0	1	0	0	0	0	1	0	healthy
88	0	1	0	1	0	0	1	1	1	0	healthy
103	1	1	1	1	1	1	1	1	0	1	cancer
105	1	0	0	1	1	1	0	1	0	1	cancer
117	1	0	1	0	1	0	0	1	1	0	healthy
118	1	1	0	1	1	1	1	1	0	1	cancer
121	1	1	1	0	0	1	1	0	0	0	healthy
126	1	0	1	1	1	1	0	1	0	1	cancer
128	1	0	1	0	1	1	1	1	0	1	cancer
132	1	1	1	1	1	1	1	1	0	1	cancer
136	1	0	1	0	1	1	1	1	0	1	cancer
143	1	0	1	1	0	0	0	1	1	0	healthy
148	1	0	1	1	1	1	1	1	0	1	cancer
154	1	1	1	1	1	1	1	1	0	1	cancer
157	1	0	0	1	1	1	1	1	0	1	cancer
162	1	0	1	1	1	1	1	1	0	1	cancer
164	1	1	0	1	1	1	1	1	0	1	cancer
166	1	0	0	1	1	1	1	1	0	1	cancer
169	1	0	1	0	1	1	0	0	1	0	healthy
175	1	0	1	0	1	1	1	1	0	1	cancer
183	1	0	1	1	1	1	0	1	0	1	cancer
184	1	1	0	1	1	1	1	1	0	1	cancer

3.2 Result description

The implemented genetic algorithm was applied on two different datasets, each contains train- and test data. For each dataset, 100 different parameters combination were used. Table X1 and Table Y1 show the first 50 parameter combinations (sorted by F1 score) for liver_train_1 and liver_train_2 respectively. Only the first parameters combination of each of these tables are considered because both these tables are sorted in descending order. These two best combinations were trained again for 10 times on each dataset. The results of the

repeated training is shown in table X2 for liver_train_1 and Y2 for liver_train_2. Table X2 and Table Y2 are sorted in descending order. The first parameters combination from these tables are used to test the test dataset. Table X3 and Y3 demonstrates the results on the liver_test_1 and liver_test_2 respectively.

The **best parameters combination** obtained from training the developed genetic algorithm on liver_train_1 is:

- number of Iteration: 75
- Population size: 100
- Mutation probability: 0.2
- Tausch: 5
- Classifier size: 8

This parameters combination can be found in Table X2. It can be seen from Table X3 that there are 5 false predictions.

The **best parameters combination** obtained from training the developed genetic algorithm on liver_train_2 is:

- Number of Iteration: 50
- Population size: 150
- Mutation probability: 0.1
- Tausch: 3
- Classifier size: 8

This parameters combination can be found in Table Y2. It can be seen from Table Y3 that there are 4 false predictions.

3.3 Run time measurements

Runtime for optimized Parameter Set:

->Liver1: 3:40 min

->Liver2: 4:00 min

Note: The reason why it takes so long is that the algorithm is going every iteration and there is no termination that tells when it does no improve.

4.Conclusion

Liverdata_1:

As Table X1 is sorted in descending order, it can be seen that for having good results, the data should contain at least 6 miRNAs. With the 100 combinations it is clear that the parameters: Mutation probability, Number of Iteration and Tausch have no great effect on the F1 score. It is good if the population size is bigger than 100.

hsa-miR-106a and hsa-miR-221 are always present. Hsa-miR-106 is almost always up-regulated, and hsa- miR-221 is almost always down-regulated. It is important to mention that because of randomness various miRNAs are selected when the best parameters combination of dataset_1 are train for extra 10 times.

The best classifier is:

[NOT hsa-miR-100, hsa-miR-224, hsa-miR-93, NOT hsa-miR-422a, hsa-miR-106b, NOT hsa-miR-106a ,hsa-miR-221, NOT hsa-miR-99a]

Liverdata_2:

As Table Y1 is sorted in descending order, it can be seen that for having a good result, the data should contain at least 7 miRNAs. With the 100 combinations it is clear that the parameters: Mutation probability, Number of Iteration, Population size and Tausch have no great effect on the F1 score.

Since there are so many different miRNAs that are randomly selected, it is difficult to decide which miRNA predicts very well. But with the 10 tests you can conclude that a down-regulated hsa-miR-199a* will give a good prediction.

The best classifier is:

[NOT hsa-mir-199a*, NOT hsa-mir-196b, hsa-mir-30d, hsa-mir-188, hsa-mir-301 , NOT hsa-mir-199b, hsa-mir-151 ,NOT hsa-mir-26a]

5. Attechments

Implementation: https://github.com/anthonyhami/SWP

6.References

https://www.tutorialspoint.com/genetic_algorithms/index.htm (30.05.2020)

https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9(25.05.2020)

PDF “INTRODUCTION TO GENETIC ALGORITHMS”

PDF “ INTRODUCTION TO CELL CLASSIFIERS”