An Efficient Hybrid Classifier for Cancer Detection

https://doi.org/10.3991/ijoe.v17i03.19683

Yangyang Chang (✉)
University of Minnesota Twin Cities, Minneapolis, USA
chan1729@umn.edu

Fadi Abu-Amara
Higher Colleges of Technology, Sharjah, United Arab Emirates

Abstract—The early detection of cancer in both healthy and high-risk populations offers increased opportunity for treatment and curative intent. In this paper, we propose a hybrid classifier that produces an efficient classification system for cancer detection in cell datasets. The first part of this work investigates the performance of artificial neural networks (ANN) such as Self-Organizing Feature Map (SOM) and Learning Vector Quantization (LVQ), while in the second part, we present our investigation on the performances of Decision Tree (DT) and its pruning model. We also, in the third part, present our proposal for a new hybrid classifier that is based on the Random Forest (RF) and the combination of the LVQ and DT. Experimental results of the proposed hybrid classifier indicate that the hybrid classifier effectively avoids the drawbacks of individual classifiers and has high anti-noise performance.

Keywords—Self-Organizing Map, Learning Vector Quantization, Decision Tree, Cancer Detection, Hybrid Classifier, Bootstrap Sampling

1 Introduction

The use of artificial intelligence in biomedical engineering includes three phases; sensor input, signal processing, and classification. Few classifiers, such as the Decision Tree (DT) and Learning Vector Quantization (LVQ), have a near-optimal performance for different databases [4, 17, 28]. However, only a few classifiers used in biomedical databases where their performances depend on the used database and special operation conditions [3, 6, 18]. On the other hand, new methods were proposed to improve the performance of individual classifiers such as the Genetic Algorithm [15], Hybrid Genetic Algorithm [15], and Swarm Optimization [26, 27], to list a few. However, the improved classifiers still have inherent flaws from their basic algorithms and complicated coding issue with, in many cases, lack discussion of classification time. The Self-Organizing Feature Map (SOM) represents a simple neural network-based classifier, Kohonen model-based, that utilizes the principle of competition between neurons [2, 3]. The SOM was developed based on the human brain neurons' function [1]. Unsupervised learning classifiers, such as SOM, are evolved into supervised learning classifiers such as the LVQ [23]. Also, other supervised learning classifiers such as the DT were developed based on Iterative Dichotomiser 3 (ID3), C4.5, or the Classification and Regression Trees [7, 22]. The study of Decision Tree is utilized to develop the Random Forest (RF), a combination of multiple Decision Trees, that is commonly used in ensemble methods to avoid overfitting [10]. In this paper, we investigate unsupervised classifiers, such as the SOM, to analyze the performance of Artificial Neural Networks (ANN) with a competition layer. Also, we investigate the performance of supervised-learning classifiers such as LVQ, DT, and RF. This paper aims at proposing an efficient hybrid classifier, utilizing the ensemble method, based on results from different investigated classifiers for the biomedical databases. Each classifier is analyzed through realigned training and testing dataset from the breast cell database of the UCI machine learning repository [19]. The proposed hybrid classifier should overcome the disadvantages of individual classifiers and provide high classification performance on different databases.

2 Cell Tumor (CT) Database

Under the microscope, images of tumor cell disclose important information related to the possibility of having cancer. The database from the University of Wisconsin includes 357 benign cases and 212 malignant cases [19]. Features of a typical cell from the database include radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension [20]. These ten features are extracted from a digitized image of a fine needle aspirate of a breast mass. The extracted features describe the characteristics of each cell nucleus found in the tumor image. Also, the average, standard deviation, and worst value of the ten characteristics are calculated. For each case, the database has thirty extracted features; the average values of ten features form the 1st to 10th features, the standard deviation of ten features forms the 11th to 20th features, and the worst values from the 21st to 30th features. Table 1 illustrates the used equations to calculate the ten characteristics of cell images where $\mu = 1/N\,(p\_1 + p\_2 + ..\, p\_N\,)$, p_i is the gray-scale value, N is the number of pixels in the moving window, E is the ratio of number of pixels to the real scale of the cell, q is number of pixels on cell covered area of tumor cell image, h is number of points on the perimeter, i is the point serial number, θ_2 is the coordinate of centroid of the tumor image, $\vec{P_i}$ is the coordinate of one point on the perimeter of the tumor cell image, $P_1, P_2,$ and P_3 are the initial point, point with the highest curvature, and final point; respectively, $\vec{V}$ is the direction vector of concavity, P_i is the point with the highest curvature in tumor cell image, θ_1 is the point of centroid of symmetry lines from the standard Non-tumor image, and θ_2 is the point of centroid of tumor cell image.

Table 1. Cell feature selection

Feature Selected	Description	Feature Mathematical Computation
Texture(f)	Estimated by the standard deviation of gray-scale values	$\mathrm{f} = \mu \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i-\mu)^2}$
Perimeter(P)		$\mathrm{P} = \mathrm{E}\sum_{i=1}^{w} p_i$
Area(A)		$\mathrm{A} = \mathrm{E}\sum_{i=1}^{q} p_i$
Radius(R)	Radius	$\mathrm{R} = \frac{\sum_{i=1}^{h}(\lvert\vec{P}_i-\overrightarrow{\theta_2}\rvert)}{h}$
Smoothness	Local variation in radius lengths	$\mathrm{S} = \sqrt{\sum_{i=1}^{N}(p_i-\mu)^2}$
Compactness		$\frac{Perimeter^2}{Area-1}$
Concavity	Measured by the angle in radians of the severity of concave portions of the contour	$\angle\vec{V} = \overrightarrow{P_1P_2+P_2P_3}$
Concave Points	Number of concave portions of the contour	C= $\sum_{i=0} P_i$
Symmetry	Estimated by calculating the centroid of the bilateral Symmetry lines from Non-tumor images and curve asymmetry lines from tumor cell images	$\angle(\overrightarrow{\theta_1}-\overrightarrow{\theta_2})$
Fractal Dimension		Coastline Approximation - 1

3 Artificial Neuron Network with Competition Layer

This section discusses the competition layer ANN as a mathematical model, theoretical basis, and simulation steps. The first part of this section discusses the unsupervised learning competition SOM, while the second part of this section discusses the supervised learning classifier LVQ.

3.1 Self-Organizing Map (SOM)

The classification idea is based on the principle that any external stimuli cause changes to the internal neuronal parameters rather than neuron's position [3]. This change will generate a specialized tissue as a reaction to the external stimuli. As the external stimuli keep increasing, the internal parameters' change continues to increase, which generates a clustering function. For near and local neurons' interaction in the self-organizing cluster, some studies expressed this interaction as a neural lateral interaction [3]. Within the circle center of the neuron that sends messages, the adjacent neurons are excited while the far neurons are in the passive state.

For the mathematic model of SOM [1], the j^{th} neural input in the competition layer is expressed by (1):

$$I_j = \sum_i W_{ij} X_i \tag{1}$$

where X represents the external signal and W_{ij} represents the weight parameters between the competition layer neuron and input layer. The inner product is shown in (2).

$$||X - W_e|| = \min_{i=1} \left||X - W_j|\right| \left||X - W_j|\right| = \sqrt{\sum_{i=1}^{n} \left(X_i - W_{ij}\right)^2} \tag{2}$$

The j^{th} output of the competition layer can be described as shown in (3).

$$dY_j/dt = I_j + \sum_{k \in S_j} R_k - g\left(Y_j\right) \tag{3}$$

where S_j represents the group of near neurons, R_k represents the weighted parameters between near neurons, and $g\left(Y_j\right)$ represents some nonlinear lose. If the j^{th} neuron is wining the specific stimulus, the output will be one, Y_j is equal to one, and the competition layer will be the output layer. Because of the interaction, the weighted parameters will change per stimuli, as shown in (4), and thus there must be a group of neurons that respond to the stimuli trigger.

$$dW_j/dt = \alpha Y_j X - \mathrm{B} Y_j W_j \tag{4}$$

where W_j represents the weighted vector, X represents the input vector, and α and B represent the adjustment number. If the N_c represents the winning group of neurons, the output Y_j in the group N_c should be one, B will be α, which will lead to the same stimuli, and the weighted parameters will be decreased in value. If the Y_j did not change, in this group, no matter how many times the stimuli trigger, the Y_j is called N_c beyond doubt. Also, if the Y_j is not in the group, the B and Y_j will be 0, indicating that the parameters will not change, as shown in (5). This procedure is captured in the learning process of SOM.

$$\begin{cases} \frac{dW_j}{dt} = \alpha\left(X - W_j\right), & j \in N_c \\ \frac{dW_j}{dt} = 0, & otherwise \end{cases} \tag{5}$$

At the end of the learning process, α represents the study rate, which is a gradually decreasing function with only one variable; study time. Therefore, α may be represented by any function such as $\alpha = 1/\mathrm{t}$ where t is the study time. The MATLAB offers a basic programming mode of setting the competition network [10]. In this article, for the first simulation, the SOM is used to completely classify 100 different patients randomly extracted from the cell tumor (CT) database, which can deeply find the performance of ANN with the competition layer. Since there are 100 different patients, the number of neurons is set to 100. For the second simulation, the SOM is used to classify two cell categories; malignant and benign.

One can state that three key parameters can affect the performance of the ANN with competition layer; learning / iteration time, number of neurons in the completion layer, and study rate as indicated in Ref. [5]. In the first simulation, 600 training samples and 168 testing samples, the learning/iteration time will change gradually as 10, 20, 50, 100, 250, 500, and 1500 to find the suitable iteration while the study rate is set to 0.01 to obtain fast and accurate classification [1, 13]. In the second simulation, the suitable iteration is kept fixed to find a suitable study rate where the SOM changes the number of neurons to 2 to classify two cell categories. The testing database for the second simulation consisting of 569 cases from the CT Database. Figure 1 shows the flowchart of designing the SOM. There are two tests in the whole process. In the first test, the SOM should classify 100 different patients. In the second test, the SOM should classify two categories of 569 patients.

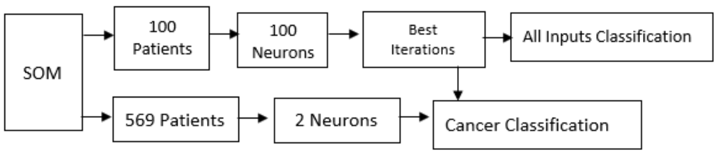


Fig. 1. Flowchart of the proposed SOM

3.2 Learning Vector Quantization (LVQ)

The LVQ is developed based on the Kohonen Competition Function, which has the same principle of SOM [23]. The main difference between the two networks is the utilization of LVQ on supervised learning. The LVQ network is composed of three different layers; input, competition, and output, so the LVQ can automatically adjust the weighted parameters to obtain correct classification at the output layer and sets the weighted parameters between the competition layer and output layer. The values of these weighted parameters are always one or zero since the wining neurons should be one when it is associated with a particular class of input, while zero indicates failing

neurons. The winning neuron from the competition layer is decided by (2) where the process of updating the parameters is similar to (5). Still, we consider the runner-up neuron in the case of the wrong winning neuron and the restricted decision by one neuron [13]. Comparing the output layer result with actual classification, the LVQ can update the weighted parameter without many iterations to ensure better performance by setting the suitable mean square error (MSE) for training [14].

To find the suitable MSE and iteration of the proposed Advanced LVQ (ALVQ) classifier, the value of MSE is set to the maximum value from five individual Fast-Good LVQ classifiers with the largest iteration. The requirements of the Fast-Good LVQ are set to: classification time less than 10 seconds, and accuracy, sensitivity, and specificity are larger than 85%.

Figure 2 shows the flowchart of designing the ALVQ classifier. First, this paper manually finds valid ranges of MSE, iteration, and number of neurons for Fast-Good LVQs. Second, the Five LVQs take 150% of the maximum iteration time with the minimum number of neurons from the previous results. Third, the ALVQ utilizes the maximum MSE with associated iteration from Five LVQs. The number of neurons of

ALVQ is set to the maximum number of the Fast-Good LVQs. In this section, the same CT Database that consists of 569 cases, 519 training cases, and 50 testing cases is used. To evaluate the performance of classifiers, the database will be randomly realigned ten times before each simulation.

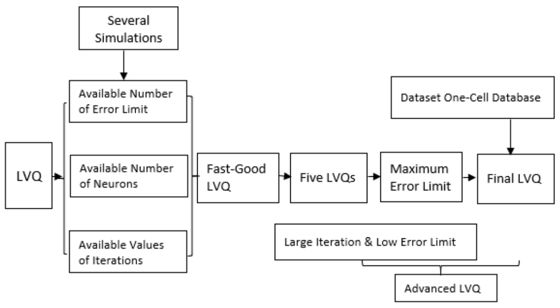

Several Simulations
Available Number of Error Limit
Available Number of Neurons
Available Values of Iterations
LVQ
Fast-Good LVQ
Five LVQs
Maximum Error Limit
Final LVQ
Dataset One-Cell Database
Large Iteration & Low Error Limit
Advanced LVQ

Fig. 2. Flowchart of the proposed ALVQ Classifier

To evaluate the anti-noise classification performance, this paper uses ALVQ to conduct another three tests. The first test employs random Gaussian noise into 50% of the training set. The second test adds an unknown noise into a quarter of the training set through three random distributions with random parameters. The third test changes the data of 20 patients in the training set into a different value. This process uses the Colon Cancer Database. The whole simulation is conducted ten times due to random parameters of the noise.

The influence of iteration and number of neurons in the competition layer is analyzed in the SOM Section. Since the SOM doesn't have the output layer and competition layer, the SOM classifier performance will not be affected by supervised teaching. We can easily find the effects of iteration and the number of neurons, with a fixed study rate, on the results without setting the MSE of LVQ. The results indicate classification improvement in the learning function and study rate in comparison with other results such as [13, 25]. This paper uses a 0.01 study rate, which is enough to make an accurate classification for many small databases within ten iterations if there is no significant noise in the training set or insufficient training process [13].

4 Decision Tree

The DT is considered as a statistical technique of the information reaction that is based on the entropy and information gain from the ID3 algorithm [7]. The entropy $H(V)$ of information is used to describe the purity of group information, as shown in (6). If all samples are in the same group, the $H(V)$ will be zero. In other words, if a system has mostly the same information, its entropy will be minimal. The DT is constructed with each non-terminal/non-leaf node representing the selected attribute on which the data is split. Further, terminal/leaf nodes represent the class label of the classification decision after computing all attributes [24]. In ID3, the entropy always affects the non-leaf node. The smallest entropy or largest information gain is used to be the non-leaf node in every iteration. The larger the number of single attributes in the data set, the higher probability of this attribute becoming a non-leaf node. To avoid drawbacks of the ID3 of tending to adopt more samples in the classifier where many tree branches offer useless details, the C4.5 algorithm is used where the gain is replaced by the gain ratio, as shown in (6), [8].

$$IG\_ratio = IG(V)/H(V), H(V) = -\sum_j p(v_j) \log_2 v_j \qquad (6)$$

where IG_{ratio} represents the information gain, v_j represents the difference value from the group feature V, and $p(v_j)$ represents the proportion of sample j in group V.

Furthermore, due to the binary categories of the CT database, the Classification and Regression Trees (CART) is a better choice [12]. The CART implementation is similar to the C4.5. The only difference is that CART is based on the Gini index, which is a standard impurity measure [22]. The CART can generate a regression tree. However, this work focuses on classification by the CART Method. To avoid the overfitting issue, this work uses the pessimistic pruning algorithm [9]. The pessimistic pruning method replaces the previous non-leaf node with the leaf node. To maintain the information classification accuracy and efficiency, the pessimistic pruning algorithm bases its judgment on the classification error. For a leaf node, it covers N samples, and the ratio of classification error is (E+0.5)/N. If the inside non-leaf node has L following nodes, the ratio of classification error can be found, as shown in (7).

$$\mathrm{e} = \left(\sum \frac{\mathrm{E_i} + 0.5 * \mathrm{L}}{\sum \mathrm{N_i}}\right) \qquad (7)$$

where the constant 0.5 represents the punishment of calculating error ratio for a specific node. In the case of two-group classification, the error follows the Bernoulli Distribution. The calculation of the standard deviation of the error of a sub-node inside a non-leaf node can be found using (8).

$$\sigma\,(\mathrm{subtree}) = \sqrt{\mathrm{N} * \mathrm{e} * (1 - \mathrm{e})} \qquad (8)$$

Assuming the final leaf node will be cut, and the error classification is J, then the ratio of error should be (J+0.5)/N. If the difference between the non-leaf node error ratio and sub-node error ratio is larger than the signal node, the tree branch will be cut.

Figure 3. shows the flow chart of the DT used in this paper. The DT uses the same CT database used by LVQ. To ensure finding the correct performance, the database will be randomly realigned ten times before each classification. Also, the mathematical logic procedure and the confusion matrix will be used to investigate the performance of the pruning tree from the CART and the normal DT from CART. Finally, the same anti-noising tests applied to LVQ will be applied to the Pruning CART.

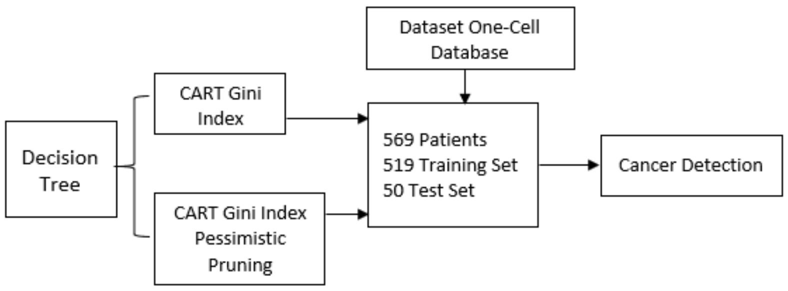


Fig. 3. Flowchart of the proposed supervised classifier

5 Hybrid Classifier

A hybrid classifier is proposed based on the random forest (RF) and the combination of LVQ and DT. The purpose of the hybrid classifier is to overcome the drawbacks of each classifier based on the analyzed performance of individual classifiers.

From the mathematical inference of ANN with the competition Layer, the LVQ must take enough iterations to adjust its weighted parameters. However, not all neurons in the competition layer can make accurate reactions to different inputs. In the case of too large/small Euclidean distances between some vectors of input data, the learning process of competition and random initial weighed parameters limits the classification accuracy. In this case, the LVQ utilizes this drawback to avoid a big noisy training set. However, the DT can be more sensitive to any perturbations in the training data making it hard to avoid a wrong classification for noisy training data. Therefore, the LVQ can make up the drawbacks of DT. Furthermore, the DT can make up the deficient performance of neurons. When considering the multicollinearity drawback, the DT greedily chooses the significant group feature, and the LVQ chooses the small Euclidean distance group. Finally, ensemble methods, such as RFs, can negate this issue.

The RF is considered as multiple DTs-based classifier where the output category is determined by the output mode of individual trees [10]. When data progress inside the RF, it cause each DT to categories the data. The RF makes the final decision based on the majority vote of individual DTs. The Bootstrap Sampling method is used where it samples the data to create training groups equal to the number of used DTs inside the RF [21]. If a group S contains n samples x_1, x_2… ,x_n, then sampling the group n times creates a new group S'. For the new group S', the probability of not including one sample is $= (1 - 1/n)^n$. Thus, we can get a new group with the same number of samples but different from the previous one. The new group S' is created as a training group for a specific DT [11]. The C4.5 method is fast enough to run one hundred Decision Tree models. In summary, we used 100 new groups as 100 training sets for 100 Decision Tree models with the C4.5 method. The classification result is based on the majority vote from DT results. Meanwhile, inherited problems of RFs are incomprehensible and hardly incremental in the classification process. The DT and LVQ can provide an inside process of classified features that should efficiently make up the flaws of RFs.

The proposed hybrid classifier consists of two sub-hybrid classifiers, where its framework is presented in Figure 4. The first sub-hybrid classifier bases its classification decision on two classifiers. The LVQ-based classifier uses the ALVQ with suitable training MSE and iteration from the Five LVQs. The DT-based classifier uses the pruning CART method. The other sub-hybrid classifier employs the RF where one hundred C4.5 DTs, based on the Bootstrap Sampling method, are used to create new one hundred training sets from the 519 training cases.

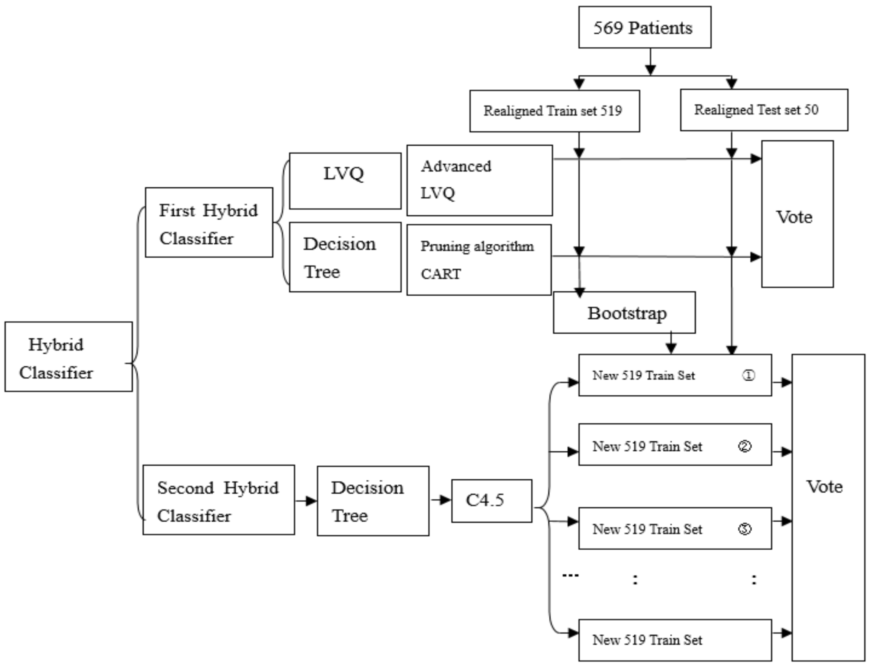


Fig. 4. The flowchart of the proposed hybrid supervised classifier

To reduce the classification time, the three classifiers are run in parallel to vote for the final classification result. The hybrid classifier performance is also tested against noisy cases in the same test setup of the LVQ and DT individual classifiers. The RF is considered as multiple DTs-based classifier where the output category is determined by the output mode of individual trees [10]. When data progress inside the RF, it cause each DT to categories the data. The RF makes the final decision based on the majority vote of individual DTs.

6 Experimental Work and Performance Analysis

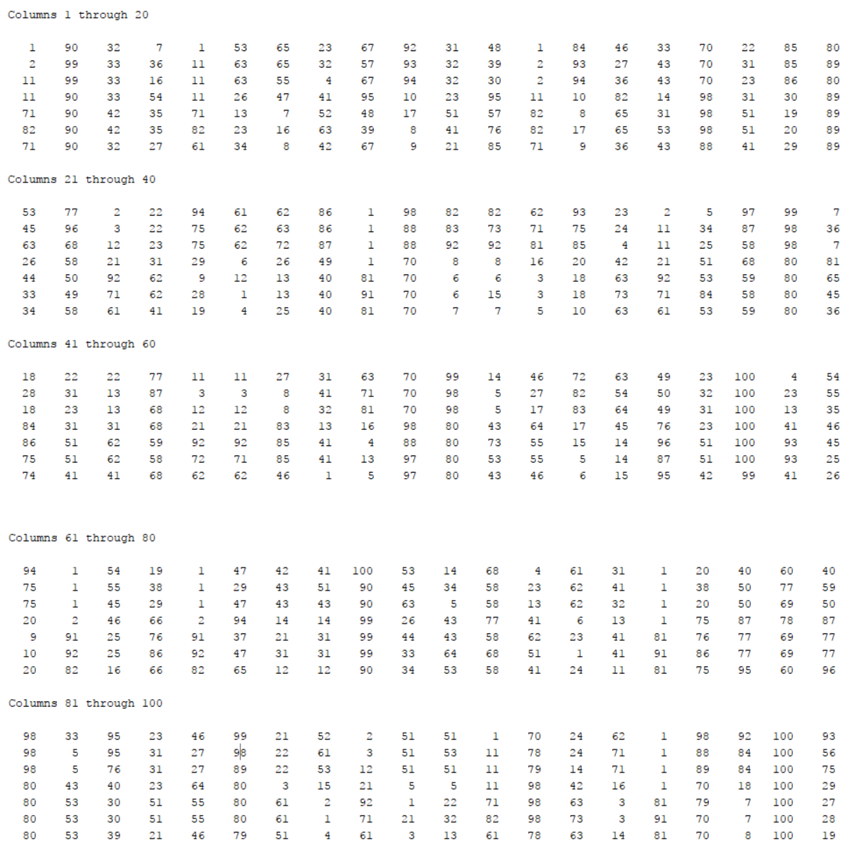
This section presents the simulation results of three individual classifiers and the proposed hybrid classifier.

6.1 Classifier performance evaluation

The accuracy, sensitivity, and specificity are used to evaluate classifiers' performance, as shown in (9).

$$\text{Accuracy} = \frac{(\text{TP+TN})}{(\text{TP+FN+FP+TN})}\ \text{Sensitivity} = \frac{\text{TP}}{(\text{TP+FN})}\ \text{Specificity}\ \frac{\text{TN}}{(\text{TN+FP})} \tag{9}$$

where TP indicates true positive, TN indicates true negative, FN indicates false negative, and FP indicates false positive.



```
Columns 1 through 20

     1    90    32     7     1    53    65    23    67    92    31    48     1    84    46    33    70    22    85    80
     2    99    33    36    11    63    65    32    57    93    32    39     2    93    27    43    70    31    85    89
    11    99    33    16    11    63    55     4    67    94    32    30     2    94    36    43    70    23    86    80
    11    90    33    54    11    26    47    41    95    10    23    95    11    10    82    14    98    31    30    89
    71    90    42    35    71    13     7    52    48    17    51    57    82     8    65    31    98    51    19    89
    82    90    42    35    82    23    16    63    39     8    41    76    82    17    65    53    98    51    20    89
    71    90    32    27    61    34     8    42    67     9    21    85    71     9    36    43    88    41    29    89

Columns 21 through 40

    53    77     2    22    94    61    62    86     1    98    82    82    62    93    23     2     5    97    99     7
    45    96     3    22    75    62    63    86     1    88    83    73    71    75    24    11    34    87    98    36
    63    68    12    23    75    62    72    87     1    88    92    92    81    85     4    11    25    58    98     7
    26    58    21    31    29     6    26    49     1    70     8     8    16    20    42    21    51    68    80    81
    44    50    92    62     9    12    13    40    81    70     6     6     3    18    63    92    53    59    80    65
    33    49    71    62    28     1    13    40    91    70     6    15     3    18    73    71    84    58    80    45
    34    58    61    41    19     4    25    40    81    70     7     7     5    10    63    61    53    59    80    36

Columns 41 through 60

    18    22    22    77    11    11    27    31    63    70    99    14    46    72    63    49    23   100     4    54
    28    31    13    87     3     3     8    41    71    70    98     5    27    82    54    50    32   100    23    55
    18    23    13    68    12    12     8    32    81    70    98     5    17    83    64    49    31   100    13    35
    84    31    31    68    21    21    83    13    16    98    80    43    64    17    45    76    23   100    41    46
    86    51    62    59    92    92    85    41     4    88    80    73    55    15    14    96    51   100    93    45
    75    51    62    58    72    71    85    41    13    97    80    53    55     5    14    87    51   100    93    25
    74    41    41    68    62    62    46     1     5    97    80    43    46     6    15    95    42    99    41    26


Columns 61 through 80

    94     1    54    19     1    47    42    41   100    53    14    68     4    61    31     1    20    40    60    40
    75     1    55    38     1    29    43    51    90    45    34    58    23    62    41     1    38    50    77    59
    75     1    45    29     1    47    43    43    90    63     5    58    13    62    32     1    20    50    69    50
    20     2    46    66     2    94    14    14    99    26    43    77    41     6    13     1    75    87    78    87
     9    91    25    76    91    37    21    31    99    44    43    58    62    23    41    81    76    77    69    77
    10    92    25    86    92    47    31    31    99    33    64    68    51     1    41    91    86    77    69    77
    20    82    16    66    82    65    12    12    90    34    53    58    41    24    11    81    75    95    60    96

Columns 81 through 100

    98    33    95    23    46    99    21    52     2    51    51     1    70    24    62     1    98    92   100    93
    98     5    95    31    27    98    22    61     3    51    53    11    78    24    71     1    88    84   100    56
    98     5    76    31    27    89    22    53    12    51    51    11    79    14    71     1    89    84   100    75
    80    43    40    23    64    80     3    15    21     5     5    11    98    42    16     1    70    18   100    29
    80    53    30    51    55    80    61     2    92     1    22    71    98    63     3    81    79     7   100    27
    80    53    30    51    55    80    61     1    71    21    32    82    98    73     3    91    70     7   100    28
    80    53    39    21    46    79    51     4    61     3    13    61    78    63    14    81    70     8   100    19
```

Fig. 5. Classification Results of the SOM

6.2 SOM classification analysis

Figure 5 shows the classification results of the SOM. At the beginning of the simulation, each neuron in the network is given a serial number so the program can easily check which neuron is related to a specific category of the database. The columns of figure 5 represent the patient number, while rows indicate the iteration (10, 20, 50, 100, 250, 500, 1500). The intersection of patient number and iteration indicates neurons' serial number.

The SOM system assigns every neuron a random serial number in each iteration, which represents one input category. For example, in Figure 5, for the first patient when the iteration is 250, the neuron's serial number of the classifier is 71. As Figure 5 indicates, using more iterations provides the neuron with better-weighted parameters like the human's memory since it requires more time stimulus to provide better results. However, simulation results indicate that using a large iteration will not efficiently enhance classification performance anymore; on the contrary, it decreases performance because of long classification time with almost unimproved accuracy. Simulation results also indicate that using 250 iterations results in acceptable accuracy and classification time.

In the case of using iteration less than 250, the SOM correctly identifies less than seventy patients. On the other hand, using iteration larger than 250 results in a little-improved accuracy for the SOM network but at the price of longer classification time.

Results also show that using 500 iterations allows the SOM to avoid the error of 250 iterations, such as the change between 31st and 32nd columns, which takes longer classification time close to ten seconds. However, using 500 iterations introduces a new error, such as the change between 16th and 52nd columns. Figure 5 also indicates that no matter how the iterations change, the SOM cannot correctly classify five patients; number 81, 39, 51, 99, and 58. These results are due to the limited learning ability of neurons or due to disabled neurons being unable to effectively update the weighted parameters along with the iteration and study rate. In summary, the 250-iteration results in the best classification performance.

If there are more than 100 neurons to classify 100 patients in the SOM competition layer, the classification performance of SOM should produce better performance. The SOM is also used for cancer detection. The previous discussion recommends using the 250 iterations. The results, shown in Table 2, indicate the SOM performance in cancer classification. Table 2 indicates that the number of used neurons affects classification time. Figure 6 shows the reaction time of neurons to the input data. These two experiments indicate that the iteration and number of neurons largely affect ANN with the competition layer. Using more iterations can enhance classification performance but increases the classification time. Using more neurons can avoid the overfitting problem and efficiently enhance the classification performance. Comparing the SOM with the traditional K-Means method, the SOM overcomes many of the K-means drawbacks [16]. For example, in the case of the unknown database, more cases than the predicted number of classified groups in the training group must be known using the K-means classifier.

6.3 LVQ classification analysis

Figure 7 shows an example of the LVQ training confusion matrix with 20 neurons, 0.1 error limit, and 250 iterations. The true positive set has 322 patients which occupy 62% of all training sets, the false negative set has 42 patients, the false positive set has 7 patients, and the true negative set has 148 patients. Results indicate 88.5% sensitivity, 95.5% specificity, 97.9% precision, 77.9% negative predictive value, and 90.4% accuracy.

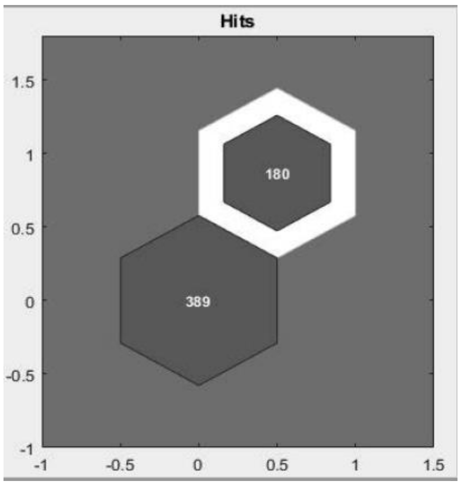
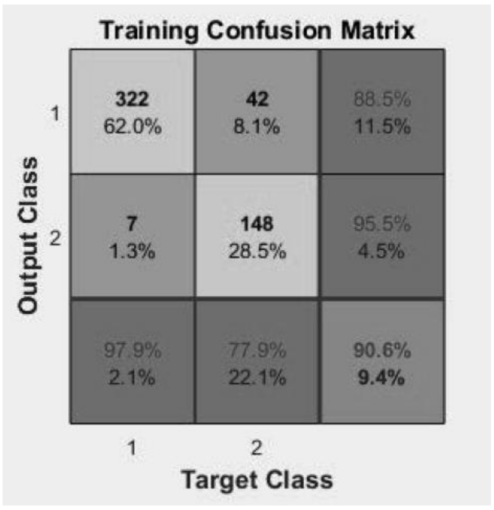


Fig. 6. Times of Neuronal Reaction



Training Confusion Matrix

Output Class \ Target Class	1	2	
1	**322** 62.0%	**42** 8.1%	88.5% 11.5%
2	**7** 1.3%	**148** 28.5%	95.5% 4.5%
	97.9% 2.1%	77.9% 22.1%	90.6% **9.4%**

Fig. 7. An example of LVQ training confusion matrix

Table 2. SOM classification performance in identifying cancer

Classification Targets		**569 Patients (Two Neurons)**	
Confusion matrix (Training group)		***Predicted***	
		Negative	***Positive***
Actual	Negative	347	10
	Positive	42	170
Performance	Accuracy	***Sensitivity***	***Specificity***
Test group	90.86%	80.18%	97.20%
Iteration times		***Simulation Time (Two neurons/ neurons)***	
<50		<10 second	
100		17 second	
250		44 seconds	
>500		90 seconds	

Fast-good LVQ results: To find a suitable MSE range, this paper designs LVQ with very small MSE and enough number of iterations. Figure 8 shows a typical example of the MSE curve versus Epochs for a training MSE of 0.01 and 250 iterations. Figure 8 shows that MSE cannot get a lower value than 0.0829 after 9 iterations. Since the supervised LVQ can efficiently adjust the weighted parameters of the neuron from the actual output, the LVQ takes the shortest time of iteration to achieve the best performance. The LVQ cannot reach 0 MSE due to the disabled neurons and the drawback of competition learning processing, which already indicated in the SOM Section. After several simulations with the random training data and testing data, Table 3 shows the MSE training goal range to reach Fast-Good LVQ. Table 3 indicates that the LVQ can have the best performance within 10 iterations and 10 seconds for the realigned CT Database. Figure 8 indicates that the performance will be distorted when LVQ cannot get 0.0829 MSE and continues to classify until 250 iteration ends. In this case, the number of neurons should be chosen from 20 to 25 because the simulation time of 30 neurons is close to 10 seconds, and a smaller number of neurons can inherent the overfitting problem as discussed in the SOM Section. So, the Fast-Good LVQ should have MSE less or equal to 0.1, 15 to 30 neurons, and 10 iterations.

Table 3. LVQ classification performance in identifying cancer

Condition	**Range Best MSE**	**Iteration**	**Time**
10~30 Neurons	0.0925~ 0.082852	<10	<10s
Performance	***Accuracy***	***Sensitivity***	***Specificity***
	>90%	>85%	>92%

The ALVQ results: From the analysis of the previous section, parameters of the five LVQs of the ALVQ should have 0.01 MSE, 15 iterations, and 10 neurons. The MSE and iteration of the ALVQ are found from five LVQs. This accurate classifier can automatically find the suitable weighted parameters from five ALVQs, and it can adapt to other databases with a similar scale. Table 4 shows the ALVQ classification performance in identifying cancer. As Table 4 shows, after ten times of simulation with realigned training and testing sets of the CT Database, the ALVQ has average accuracy and specificity above 90%. However, the sensitivity is around 82%, which means the ALVQ suffers the overfitting problem for some malignancy inputs.

Table 5 shows the simulation results of the anti-noise performance of the ALVQ. From the simulation results, the LVQ has high anti-noise classification performance in case of large SNR and, in case of the noise, has different distribution function. However, if we change 20 input data into 2000, the LVQ will be crashed. Figure 9 indicates that LVQ needs more classification time to handle significant different input data. Therefore, for the ALVQ, if there is a considerable noise, using short iteration time will not be enough.

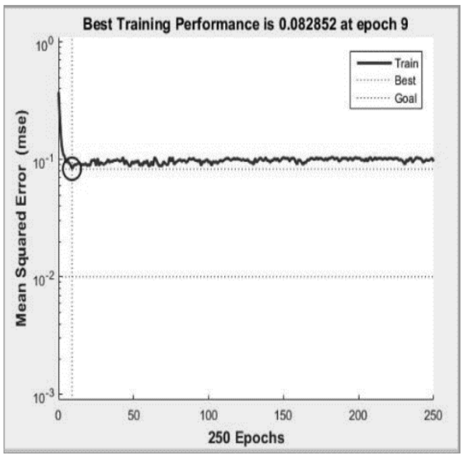


Fig. 8. Example of Curve of MSE

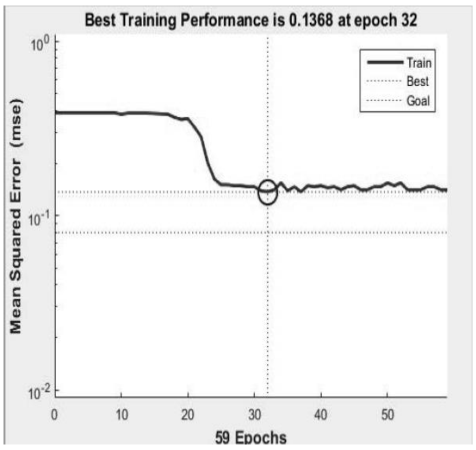


Fig. 9. MSE Curve of the third noisy testing set

Table 4. LVQ classification performance in identifying cancer

	Total number	**Benign tumor**	**Malignancy**
Number patients	569	357	212
Advanced LVQ			
Confusion matrix (Random 50 Test group)		***Predicted (Average)***	
		Negative	***Positive***
Actual	Negative	30.8	0.2
	Positive	3	16
Performance	***Accuracy***	***Sensitivity***	***Specificity***
Test group	93.6%	84.2%	99.35%
Total Simulation Time		14 seconds	

Table 5. LVQ anti-noise performance

No Noise			
	Total number	***Benign tumor***	***Malignancy***
Total patients	569	357	212
Test patients	519	40	10
Performance	Accuracy	Sensitivity	Specificity
Test group	98%	90%	100%
First Noise			
Noisy Patients		***Gaussian Noise SNR Ratio in dB***	
261		1 to 100	
Performance	**Accuracy**	**Sensitivity**	**Specificity**
Test group	98%	90%	100%
Second Noise			
Noisy Patients		***Three Random Distribution***	
131		Poisson, Rayleigh, Noncentral Chi-Square Distribution with random <10 parameters	
Performance	**Accuracy**	**Sensitivity**	**Specificity**
Test group	96.8%	90%	98.5%
Third Noise			
Noisy Patients		***Large Noise***	
20		0, 200, 2000	
Performance	Accuracy	Sensitivity	Specificity
The data of 20 patients changes to all 0			
Test group	94.8%	90%	96%
The data of 20 patients changes to all 200			
Test group	98%	90%	100%
The data of 20 patients changes to all 2000			
Test group	80.8%	4%	100%

6.4 Decision tree classification analysis

Table 6 shows the average classification performance of the DT for the realigned training and testing set from the CT Database. As table 6 indicates, the DT can be a competent classifier for early detection of cancer.

As Figure 10 shows, the DT can easily detect cancerous cells and find logical mathematical classification where x indicates a feature of the input data. For example, Figure 10 shows x23 as the worst value of the perimeter of the cell and X27 as the worst value of concavity of cell in input data which become the tree's chance nodes. If a cell has a perimeter larger than 114.65 and a concavity less than 0.1907, the specific person will have a 90.4% probability of not getting cancer.

From Figure 10, the mathematical function can be expressed by the DT in the type of $\mathrm{Y} = \mathrm{f}(\mathrm{X})$; like piecewise functions. For example, when $\mathrm{Y} = 2$, then $\mathrm{X}_{23} > 114.65$, $\mathrm{X}_{27} > 0.1907$, and $\mathrm{X}_{23} > 15.665$. However, it cannot build the consecutive relationship between two attributes in the type of $\mathrm{X}_{\mathrm{m}} + \mathrm{X}_{\mathrm{n}} < \mathrm{constant}$ where X is the input and (m, n) indicates the serial number of input features. This is because all X-boundary judgment is parallel with the axis. So, the DT offers straightforward judgment, but it is hard to render the mathematical relationship between each attribute with one mathematical equation.

From Figure 10, there is a degradation in the classification performance when the classifier executes in step X_{22}. When the group size is small, the judgment becomes useless due to overfitting issue. Therefore, it is essential to stop the tree from growing too deep accurately. One solution is by using the pessimistic pruning method.

The pessimistic pruning method-based DT is shown in Figure 11. The simulation results of the 3rd best-level, the 0.0289 new re-substitution error after pruning, and 0.0193 re-substitution error before pruning, indicate that the best level is three, which is the smallest value in subtrees because the DTM is based on the minimal cost calculation. The re-substitution error indicates re-inputting the data and checking the classification error. For example, for 30 leaves DT with 1000 samples having 10 error samples, the re-substitution error will be 10/1000=1%. After applying the pruning process, the substation error is increased from 0.019 to 0.029. However, as shown in table 7, the simulation performance on the testing data is better than the previous results of DT because the pruning method can effectively handle the overfitting problem.

Table 6. Decision tree classification performance in identifying cancer

		Total number	**Benign tumor**	**Malignancy**
Number of patients		569	357	212
Confusion matrix (Random 50 Test group)		***Predicted (Average)***		
		Negative	***Positive***	
Actual	Negative	28.6	1.4	
	Positive	3.4	16.6	
Performance		***Accuracy***	***Sensitivity***	***Specificity***
Test group		90.4%	83%	95.33%
Classification Time			1.2 seconds	

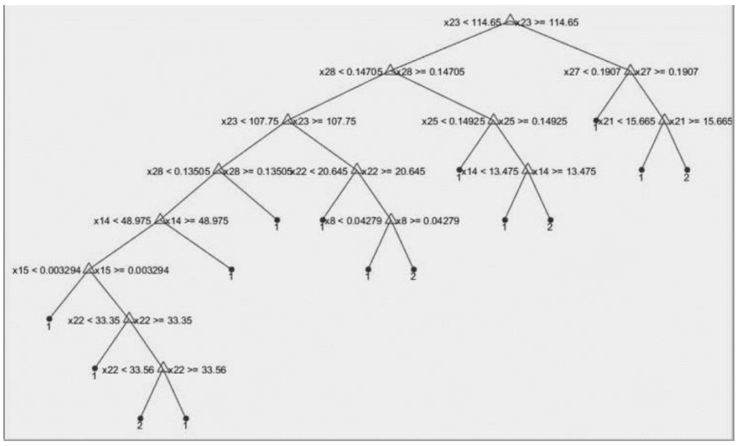

x23 < 114.65 x23 >= 114.65
x28 < 0.14705 x28 >= 0.14705
x27 < 0.1907 x27 >= 0.1907
x23 < 107.75 x23 >= 107.75
x25 < 0.14925 x25 >= 0.14925
x21 < 15.665 x21 >= 15.665
x28 < 0.13505 x28 >= 0.13505
x22 < 20.645 x22 >= 20.645
x14 < 13.475 x14 >= 13.475
x14 < 48.975 x14 >= 48.975
x8 < 0.04279 x8 >= 0.04279
x15 < 0.003294 x15 >= 0.003294
x22 < 33.35 x22 >= 33.35
x22 < 33.56 x22 >= 33.56

Fig. 10. Complete tree model

Table 7. Pruning decision tree performance

Confusion matrix (Random 50 Test group)		**Predicted (Average)**	
		Negative	***Positive***
Actual	Negative	31	0.8
	Positive	2.2	16
Performance	***Accuracy***	***Sensitivity***	***Specificity***
Test group	94%	87.91%	97.48%
Time of simulation		2.2 seconds	

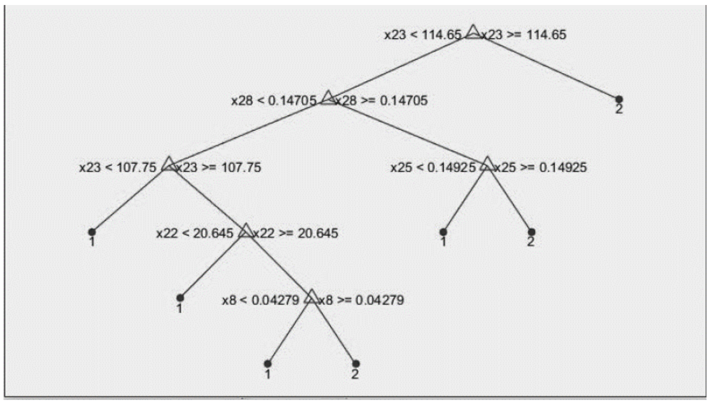

x23 < 114.65 x23 >= 114.65
x28 < 0.14705 x28 >= 0.14705
x23 < 107.75 x23 >= 107.75
x25 < 0.14925 x25 >= 0.14925
x22 < 20.645 x22 >= 20.645
x8 < 0.04279 x8 >= 0.04279

Fig. 11. Pruning the Decision Tree

Table 8 shows the simulation results of the anti-noise performance of the pruning DT. From the simulation results of the first noisy and the second noisy testing sets, the DT shows the worst performance when the noise embedded into the training set. Comparing with the LVQ anti-noise performance, Pruning DT is easily affected by noise. When the 20 patients are directly changed into 200, the pruning DT has 100%

classification accuracy for the testing group. Though the DT is pruned, the pruning DT also has the overfitting problem and makes harder classification when the small noise results in a highly correlative data. The results of the third noise test also indicate that adding suitable noise into input data, pruning the DT model shows better results.

Table 8. Decision Tree anti-noise performance

No Noise			
	Total number	***Benign tumor***	***Malignancy***
Total Patients	569	357	212
Training Sets	519	317	202
Test Sets	50	40	10
Performance	Accuracy	Sensitivity	Specificity
Test group	94%	90%	95%
First Noise			
Noisy Patients		***Gaussian Noise SNR Ratio in dB***	
261		1 to 100	
Performance	Accuracy	Sensitivity	Specificity
Test group	93.2%	90%	94%
Second Noise			
Noisy Patients		***Three Random Distribution***	
131		Poisson, Rayleigh, Noncentral Chi-Square Distribution with random <10 parameters	
Performance	Accuracy	Sensitivity	Specificity
Test group	92%	90%	92.5%
Third Noise			
Noisy Patients		***Large Noise***	
20		0, 200, 2000	
Performance	Accuracy	Sensitivity	Specificity
The data of 20 patients changes to all 0.			
Test group	92.8%	90%	93.5%
The data of 20 patients changes to all 200			
Test group	100%	100%	100%
The data of 20 patients changes to all 2000			
Test group	92.8%	90%	93.5%

6.5 Proposed hybrid classifier performance analysis

The hybrid classifier uses the CT database that consists of realigned 569 cases split into 519 training cases and 50 testing cases. The previous analysis and simulations highlight the drawbacks and merits of each classifier. Therefore, a hybrid classifier is proposed to avoid all drawbacks of individual classifiers. As Figure 12 shows, the ALVQ classifier (3rd row) adopts the suitable training MSE with the iteration from five LVQs, the DT (1st row) adopts pruning CART, and the RF classifier (2nd row) uses 100 decision trees. As Figure 12 shows, the 2nd or 9th columns make up the flaws of disabled neurons and/or the insufficient iteration, the 28th column makes up the random and/or repeating choice of uninformative input data/variable from RF, and the 40th column make up the overfitting problem of Pruning Decision Tree.

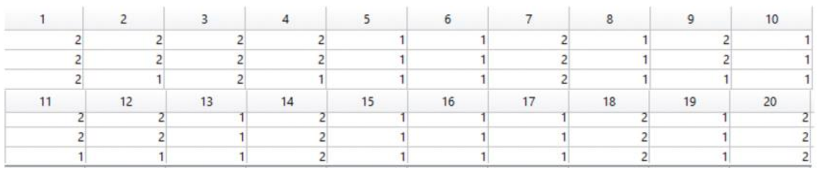
As table 9 shows, the hybrid classifier has the highest anti-noise performance. The LVQ and the RF can make up the flaws of low anti-noise performance of Pruning DT for the 1st noise. The DT and RF can make up the flaws of LVQ for the 3rd noise. However, the hybrid classifier did not have a high performance for the second noise due to the wrong votes from LVQ and Decision Tree. This drawback can also be found in table 5 and table 8. The low noise results in correlated variables of CT database, therefore, the LVQ has low ability to recognize them, and DT is easily overfitted. The DT in the hybrid classifier shows the important variables/features of input data that affects the classification and can help in viewing the logical mathematical classification. The hybrid classifier can be efficiently coded to show vectors of the training input data with LVQ weighted parameters. However, the ability of LVQ is limited by the other two DTs for the same weighted vote. If the DT gets the wrong results and LVQ made the correct classification, the hybrid classifier is prone to two DTs. Furthermore, the whole process increases the classification time by automatically find the best training MSE.

Table 9. Confusion matrix of hybrid classifier

No Noise			
	Total number	***Benign tumor***	***Malignancy***
Total Patients	569	357	212
Training Patients	519	317	202
Test Patients	50	40	10
Performance	Accuracy	Sensitivity	Specificity
Test group	100%	100%	100%
First Noise			
Noisy Patients		***Gaussian Noise SNR Ratio in dB***	
261		1 to 100	
Performance	Accuracy	Sensitivity	Specificity
Test group	97.2%	96%	97.5%
Second Noise			
Noisy Patients		***Three Random Distribution***	
131		Poisson, Rayleigh, Noncentral Chi-Square Distribution with random <10 parameters	
Performance	Accuracy	Sensitivity	Specificity
Test group	94.8%	92%	95.5%
Third Noise			
Noisy Patients		***Large Noise***	
20		0, 200, 2000	
Performance	Accuracy	Sensitivity	Specificity
The data of 20 patients changes to all 0.			
Test group	98%	100%	97.5%
The data of 20 patients changes to all 200			
Test group	100%	100%	100%
The data of 20 patients changes to all 2000			
Test group	98%	90%	100%

7 Conclusion

This paper focuses on enhancing the classification performance of cancer in cell database. Through analysis of three different classifiers, including Pruning DT, RF, and LVQ, this paper develops a hybrid classifier that has high classification performance for different small binary databases. Experimental results of ANN with a competition layer indicate that the SOM and LVQ are affected by the iteration, which recursively finds the best-weighted parameters between the competition layer and the input layer with a suitable study rate. The suitable training MSE and suitable iteration are the most critical parameters to obtain efficient LVQ. Furthermore, the ANN with the competition layer also has high anti-noise performance and stable classification time. The main drawbacks of ANN with competition layer are limitation by the learning process, and it is hard to classify some input vectors that have small (or large) Euclidean distance between each of them and the initial vectors of weighted parameters. The pessimistic pruning DT requires less computational complexity than ANN without the model training but has the worst anti-noisy performance because it demands independent and identical distribution of input data. Therefore, the normalization is a possible pre-processing solution for DT. The proposed hybrid classifier can be developed based on combination of other machine learning methods depending on the targeted database. For example, if image pixels are input instead of extracted features then the LVQ takes large computations and should be replaced by convolutional neural networks.



1	2	3	4	5	6	7	8	9	10
2	2	2	2	1	1	2	1	2	1
2	2	2	2	1	1	2	1	2	1
2	1	2	1	1	1	2	1	1	1

11	12	13	14	15	16	17	18	19	20
2	2	1	2	1	1	1	2	1	2
2	2	1	2	1	1	1	2	1	2
1	1	1	2	1	1	1	2	1	2

Fig. 12. Example of hybrid classifier results

8 References

[1] M. T. Hagan, “Neural network design,” Martin Hagan, 2014.

[2] T. Kohonen, and T.Honkela, “Kohonen network,” Scholarpedia, Vol. 2 No. 1, pp. 2(1):1568, 2007. https://doi.org/10.4249/scholarpedia.1568

[3] K. Suzuki, “Pixel-Based Artificial Neural Networks in Computer-Aided Diagnosis,” Artificial Neural Networks - Methodological Advances and Biomedical Applications, 2011. https://doi.org/10.5772/16084

[4] W. Yimyam, M. Ketcham, “The System for Driver Fatigue Monitoring Using Decision Tree via Wireless Sensor Network for Intelligent Transport System,” iJOE, Vol. 14, No. 10, pp. 21-39. 2018. https://doi.org/10.3991/ijoe.v14i10.7507

[5] T. Kohonen, Self-Organizing maps. Berlin: Springer, 2001.

[6] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, “Machine learning: a review of classification and combining techniques,” Artificial Intelligence Review, vol. 26, no. 3, pp. 159–190, 2006. https://doi.org/10.1007/s10462-007-9052-3

[7] J. Quinlan, "Simplifying decision trees," International Journal of Man-Machine Studies, vol. 27, no. 3, pp. 221–234, 1987. https://doi.org/10.1016/s0020-7373(87)80053-6
[8] S. Ruggieri, "Efficient C4.5 [classification algorithm]," IEEE Transactions on Knowledge and Data Engineering, vol. 14, no. 2, pp. 438–444, 2002. https://doi.org/10.1109/69.991727
[9] F. Esposito, D. Malerba, G. Semeraro, and J. Kay, "A comparative analysis of methods for pruning decision trees," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 5, pp. 476–493, 1997. https://doi.org/10.1109/34.589207
[10] L. Breiman, M. Last, and J. Rice, "Random Forests: Finding Quasars," Statistical Challenges in Astronomy, pp. 243–254, 2003. https://doi.org/10.1007/0-387-21529-8_16
[11] L. Breiman, "Bagging predictors," Machine Learning, vol. 24, no. 2, pp. 123–140, 1996.
[12] J. K. Jaiswal and R. Samikannu, "Application of Random Forest Algorithm on Feature Subset Selection and Classification and Regression," 2017 World Congress on Computing and Communication Technologies (WCCCT), 2017. https://doi.org/10.1109/wccct.2016.25
[13] [13] J.Yang, J. Zhan, and J. Zhang, "MATLAB Shen jing wang luo 30 li," Beijing: Dian zi gong ye chu ban she, 2014.
[14] M. Golz and D. Sommer, "The Performance of LVQ Based Automatic Relevance Determination Applied to Spontaneous Biosignals," Lecture Notes in Computer Science Knowledge-Based Intelligent Information and Engineering Systems, pp. 1256–1263, 2006. https://doi.org/10.1109/wccct.2016.25
[15] N. Chen, B. Ribeiro, A. S. Vieira, J. Duarte, and J. C. Neves, "Hybrid Genetic Algorithm and Learning Vector Quantization Modeling for Cost-Sensitive Bankruptcy Prediction," 2010 Second International Conference on Machine Learning and Computing, 2010. https://doi.org/10.1109/icmlc.2010.29
[16] U. A. Kumar and Y. Dhamija, "Comparative analysis of SOM neural network with K-means clustering algorithm," 2010 IEEE International Conference on Management of Innovation & Technology, 2010. https://doi.org/10.1109/icmit.2010.5492838
[17] L. Rokach, "Ensemble Methods for Classifiers," In: Maimon O., Rokach L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA, pp. 957-980, 2005. https://doi.org/10.1007/0-387-25465-x_45
[18] E. M. Dewi, E. Purwanti, R. Apsari, "Cervical Cell Classification using Learning Vector Quantization (LVQ) Based on Shape and Statistical Features," iJOE, Vol. 15, No. 2, pp. 91-98, 2019. https://doi.org/10.3991/ijoe.v15i02.9796
[19] UCI Machine Learning Repository (2018). http://archive.ics.uci.edu/ml/index.php.
[20] W. H. Wolberg, W. Street, D. M. Heisey, and O. L. Mangasarian, "Computer-derived nuclear features distinguish malignant from benign breast cytology," Human Pathology, vol. 26, no. 7, pp. 792–796, 1995. https://doi.org/10.1016/0046-8177(95)90229-5
[21] J. Shao and D. Tu, The jackknife and bootstrap. New York: Springer, 1996.
[22] Gordon, A. D., Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. "Classification and regression trees," Biometrics, Vol. 40 No. 3, pp. 874, 1984. https://doi.org/10.2307/2530946
[23] Pilevar, M. T., Feili, H., and Soltani, M, "Classification of persian textual documents using learning vector quantization," 2009 International Conference on Natural Language Processing and Knowledge Engineering, pp. 1-6, 2009. https://doi.org/10.1109/nlpke.2009.5313761
[24] Doumpos, M, "Multicriteria decision aid classification techniques," Applied Optimization Multicriteria Decision Aid Classification Methods, pp. 39-75, 2002. https://doi.org/10.1007/0-306-48105-7_3

[25] "Weka,"SourceForge,Feb-2018.[Online].Available:http://wekaclassalgos.sourceforge.net/.
[26] Lanzarini, L. C., Monte, A. V., Bariviera, A. F., and Santana, P. J, "Simplifying credit scoring rules using LVQ PSO," Kybernetes, Vol. 46 No. 1, pp. 8-16, 2017. https://doi.org/10.1108/k-06-2016-0158
[27] Taherkhani, M. and Safabakhsh, R, "A novel stability-based adaptive inertia weight for particle swarm optimization," Applied Soft Computing, Vol. 38, pp. 281-295, 2016. https://doi.org/10.1016/j.asoc.2015.10.004
[28] Erlinda MD, Endah P, and Retna A, "Cervical Cell Classification using Learning Vector Quantization (LVQ) Based on Shape and Statistical Features," iJOE, Vol. 15, No. 2, pp. 91-98, 2019. https://doi.org/10.3991/ijoe.v15i02.9796

9 Authors

Yangyang Chang is with the Department of Electrical and Computer Engineering, University of Minnesota Twin Cities, Minneapolis, USA. Email: chan1729@umn.edu.

Fadi Abu-Amara is an assistant professor at the CIS Division of the Higher Colleges of Technology, UAE. Fadi received his Ph.D. in Computer Engineering from Western Michigan University, USA, in 2010. His fields of interests include cryptography, cybersecurity, biomedical engineering, and socially assistive robotics. Email: fabuamara@hct.ac.ae

Article submitted 2020-11-20. Resubmitted 2020-12-28. Final acceptance 2020-12-29. Final version published as submitted by the authors.