Application of Genetic Programming and Genetic Algorithm in Evolving Emotion Recognition Module

Rahadian Yusuf, Ivan Tanev, Katsunori Shimohara
Graduate School of Science and Engineering
Doshisha University
Kyoto, Japan
(yusuf2013, itanev, kshimoha)@sil.doshisha.ac.jp

***Abstract*—This paper will discuss about implementation of a voting system and weighted credibility to augment evolution process of an emotion recognition module. The evolution process of the emotion recognition module is one part of ongoing research on designing an intelligent agent capable of emotion recognition, interaction, and expression. Genetic programming evolves the classifiers, while genetic algorithm evolves the weighted credibility as a modification of parallel voting systems. The experimental results suggest that the implementation of weighted credibility evolution improves the performance of training, in the form of significantly reduced training time needed.**

Keywords—evolutionary algorithm; genetic programming; genetic algorithm; emotion recognition; intelligent agent

I. INTRODUCTION

Intelligent agents capable of understanding user's emotion would improve the usability of computers. It is a paramount example of application for Affective Computing, where the focus is on computer usability, and usually related with emotion.

Designing such intelligent agents requires not only ability in recognizing emotion, but also the ability in adapting to a user. This requires the agent to be able to perform self-evolution, and therefore – it could not rely fully on static data sets. User's data would be changing, with additional data need to be acquired along with the interaction between the agent and the user.

There are several methods in emotion recognition and generally they are developed under several similar conditions. First, most research use generalization of multiple subjects instead of unique user's features. Second, many research use nonpervasive sensors or only use still images.

Our laboratory is conducting an ongoing research about designing intelligent agents capable of evolution, of emotion recognition, and of expression. This research uses a non-pervasive sensor in the form of Microsoft Kinect, which can also be used as other common activities such as gaming and teleconference. The acquired data are in time-series, and not only in the form of still images. This research also focuses on a single subject (unique user) and adapts to the specific unique user, instead of using a generalized data set.

One part of the research is evolving the emotion recognition module. Currently we have been implementing a voting system with several parallel classifiers to improve accuracy of recognition. However, this method needs a lot of classifiers to be evolved, in which each of them takes a lot of time. To reduce the time needed for evolution, we are exploring the possibility of two-step approach: first we evolve several classifiers using genetic programming, and then we employ genetic algorithms to evolve a voting system by means of evolutionary tuning of the weighted credibility for each classifier.

The objective of our research is to explore the possibility of said method, which is applying genetic algorithm in order to evolve weighted credibility for parallel classifiers, as a modification of the voting system. This paper will compare the results between using a normal voting system and using evolution on the weighted credibility.

II. RELATED WORKS

A. Affective Computing

Affective computing is a terminology coined by Picard [1][2]. According to the affective computing, improvements in emotion recognition of a user will also improves the general usability and promotes good human-computer interactions. Currently Picard is focusing on using skin conductance to recognize emotions.

B. Variety of Data for Emotion Recognition

There have been numerous research regarding emotion recognition, however they usually have limitations. One of the most common limitation is that many research use non-pervasive sensors, such as EEG (electroencephalography) or ECG (electrocardiography) [3].

Pentland [4] states that there are several methods on understanding human's emotion by recognizing basic gestures, especially body movements and responses. In example, the *activity* level usually correlates something about human's eagerness. Another example, someone who is engaged to a conversation, or has some interests in the topic would often make nodding gestures.

978-1-4799-7492-4/15/$31.00 ©2015 IEEE

C. Emotions from Other Discipline

From the point of view of psychology, there is a research on basic emotions that are understood by human disregarding culture and language. Ekman [5][6] went to secluded societies in the world and conducted research, which concluded that there are several facial expressions existed regardless of culture. Ekman used a photograph of actors making facial expressions and asked the people at those secluded or isolated societies to guess the emotion pictured at each photograph.

D. Emotion Classification

There are several types of emotion classifications, such as the circumplex model by Russel [7]. Plutchik [8] also proposed a model that, similar to Russel, categorized according emotion similarities.

Our research is using the circumplex model [9][10][11] that uses Valence (pleasantness or general mood) and Arousal (also similar to activity level) to represent emotions. So there are four categories of emotion to be detected by the system, as illustrated in Figure 1:

- Happiness (positive valence, positive arousal)
- Relaxed (positive valence, negative arousal)
- Sadness (negative valence, negative arousal)
- Anger (negative valence, positive arousal)

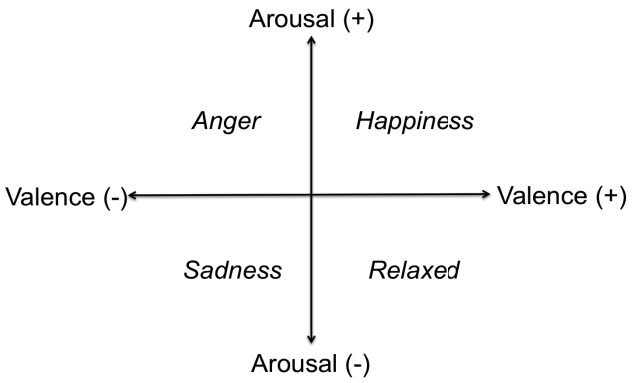


Fig. 1. Four categories of emotions

E. Proposed System

The proposed system [9][10][11] for the intelligent agent uses a non-pervasive sensor called Microsoft Kinect. It acts in a similar way to a webcam, and can be used as the perception of the system.

Figure 2 describes the general design of the intelligent agent. However, this paper will focus only on evolving an emotion recognition module.

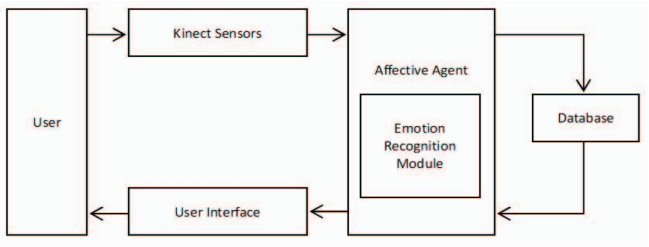


Fig. 2. Proposed system design

F. Feature Extraction

In our research we selected several features from a user's facial expression, based on the CANDIDE face model [11][12]. The Candide model represents several points in 3D face (such as the corner of a lip, the corner of left eye, etc.), and the movements of these points are represented as *Action Units*. Based on this model, our research selected 9 (nine) Action Units (AU) that correspond to the detected facial movements, which are as follows:

- Upper lip raiser (upper lip moves up or down)
- Jaw lowerer (the jaw moves up or down)
- Lip stretcher (the corners of the lip moves away or closer)
- Brow lowerer (the brow line moves up or down)
- Lip corner depressor (the corners of the lip, up or down)
- Outer brow raiser (the outer corners of the brow)
- 3 (three) Head tilt poses: yaw, pitch, and roll

The extraction of features is done in a specific time frame, as a result sampling of the flow of sensory information received form Kinect sensing device. The sampling period of the Kinect to gather a single row of raw datum (single frame) is about 33ms (gathered 100 data frames per 3 seconds). To extract the feature from the raw data, this research sample every 100 frames.

Each single row of raw datum (single frame) consists of the 9 (nine) data of AU. Our research then samples for every 100 frames to get 27 values, which are the average value, standard deviation, and power of each AU.

The average value will represent the general level of such facial movements. Standard deviation is used to represent the activity level of each AU. An integration or power of the signals is used as well as one of the extracted feature. The total number of extracted features is 27 (9x3). Each feature is extracted from 100 raw data frames (3 seconds).

G. XGP

1) XGP overview

XGP is an in-house GP engine [10][11][13][14] that features XML-based genotype representations of candidate solutions (genetic programs), XML-schema that determines the allowed syntax of the genotypes, and UDP channel to communicate between fitness evaluator and the XGPmanager. The latter manages the population of genetic programs and

performs the main genetic operations – selection, crossover and mutation, respectively.

2) Genotype representations

In our work we evolve (via XGP) the mathematical models (functions) that recognize in an optimal way the human emotions based on the extracted facial features. These models (functions) are represented by XGP as parse trees, featuring non-terminal (functional) and terminal symbols as elaborated in the following Table 1.

TABLE I. FEATURES FOR XGP

Feature Name	Values
27 extracted features (Terminal symbols)	-1,000 ~ 1,000
Random constants (Terminal symbols)	1 ~ 10
Arithmetical operators (Nonterminal symbols)	+ , - , * , /
Comparison operators (Nonterminal symbols)	< , >

3) Tree structure

The tree structure for our research is separated into two branches from the root, namely the *Valence* subtree and *Arousal* subtree. Subsequent subtrees from Valence and Arousal are in the form of functions represented by the genotype.

The following Backus–Naur Form (BNF) can represent the structure of the parse tree (i.e., the genetic representation of the recognition function), along with the syntaxes, terminal sets, and functions:

```
IF "F" "Comp" "F" THEN True
Comp::= "<"|">"
F::= "Const"|"Var"|"F","Op","Const"|"Var"|"F"
Const::= "1".."10"
Var::= "v_0".."v_29"
Op::= "+"|"-"|"*"|"/"
```

4) Fitness function

The fitness function determines the mechanism of evaluating the quality of the each individual in XGP (i.e., the evolved mathematical function for emotion recognition). In our work, we use Matthew Correlation Coefficient [15] as the fitness function, based on the accuracy and precision of prediction on valence and arousal of each data sample during training.

MCC (Matthews correlation coefficient) can be written as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Where:

TP = True positive

TN = True negative

FP = False positive

FN = False negative

MCC results in a value between -1 (exact opposite) to 1 (perfect match). The value of zero means total randomness. For the fitness function, in our system we scaled the MCC to fit within the range between 1 (perfect match) and 9999 (exact opposite). The value of 5000 indicates a total randomness.

III. EXPERIMENTS AND RESULTS

Our previous works [10][11] have performed experiments on separation of tree structure and evolving them individually, and it is shown that the method gives a better average result.

The voting system will generally improve the average of results [11]. For this paper we are exploring the possibility of another approach, which related to credibility.

The setup for XGP to evolve classifiers are similar with previous works, which are:

- Termination criteria, if # of generations reach 300, or fitness value reaches below 2, or stagnation for 32 generations
- Population size is 100 individuals
- Elite individuals are 2 individuals
- Selection rate is 10%
- Mutation rate is 3%

A. Experiment setup

The experiment is set into several configurations, and each configurations use several batches. Each batch of experiment consists of several XGP sessions, where each session will have an XGP run (random seed initiations, selections, crossovers, mutations) for several generations until termination criteria is reached.

There are two basic configurations, the genetic programming and voting configuration, and the genetic programming and credibility evolution.

Figure 3 describes the voting configuration, while Figure 4 represents the training configuration for the evolution of credibility. Our research uses a modified XGP as a GA engine.

For voting configuration, the genetic programming will evolve 10 (ten) classifiers, then the 5 (five) best of these 10 (ten) classifiers will be used for voting. Voting will be performed by using the rule of majority: if 3 (three) or more of the classifiers vote for true, then the value is true.

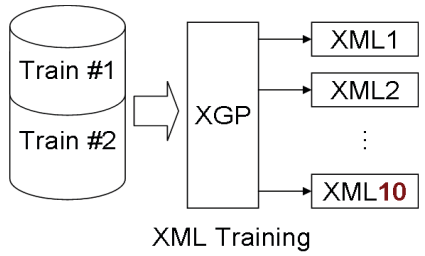
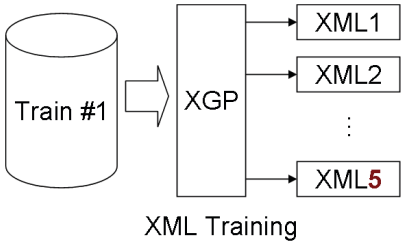


Fig. 3. Training configuration for voting



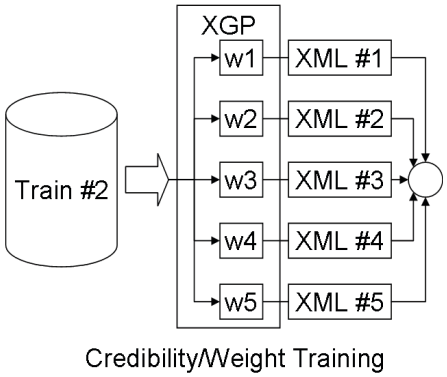


Fig. 4. Training configuration for weight credibility evolution

For configuration of weighted credibility evolution, the genetic programming will only evolves 5 (five) classifiers, which each will be multiplied by a weight evolution using genetic algorithm. This method uses autonomous learning, since no human is needed to select the better classifiers to be used.

B. *Preparations for data acquisitions*

There will be several data sets for this experiment, and this experiment makes sure that the data for training (*Train #1, Train #2*) and the data for testing (*Test*) are mutually exclusive. Further, it also makes sure that data for *Train #1* and data for *Train #2* are mutually exclusive as well, in order to obtain better objective results.

1) Data acquisitions

There are three different data acquired, but all are from a single subject. Each of the data is taken from a different time period (morning, noon, late afternoon), with different conditions regarding seat position and Microsoft Kinect's position. There is no special setup on the positions, only the required sufficient lighting condition for the Microsoft Kinect, and the general direction of the device, to operate properly. This is meant to imply the randomness of human movement in front of the camera, and also to make sure that these three sets of data are not from the same series.

2) Data classifications

The data are classified as three kinds of data: *Train #1, Train #2,* and *Test set.*

This experiment uses two methods in preparing these three kinds of data. The first one is prepared by sampling randomly and equally from each time (morning data, noon data, late afternoon data). *Train #1, Train #2,* and *Test* data sets are mutually exclusive uses. This first method is used in order to test the general ideal condition of training, by putting more combinations of data into each set. This data set is tagged as *Ideal Data Set*.

The second method is by using morning data, noon data, and late afternoon data as *Train #1, Train #2,* and *Test* data, respectively. This second method is used in order to test practical applications where it is highly inconvenient to get ideal data which represents most situation. This data set is tagged as *Practical Data Set*.

C. *Experiment Results*

There are 2 (two) configurations: *Voting* and *Credibility*. There are also 3 (three) batches of experiment for each configuration, where each batch also consisted of tree separation and combined tree evolution. Each batch consisted of several sessions, where each session consisted of several runs of XGP (which handles random seed generation, selections, mutations, and crossovers).

1) Performance (time and generations)

Evolving a single classifier using genetic programming (XGP) needs much more time and many generations compared to evolving a single weight combination. Evolving the weight only takes a maximum of 15 minutes (40 generations), while to evolve a single classifier the minimum time required is more than 15 minutes (40 generations).

Using these figures only, it is safe to state that evolving credibility is far better than evolving more classifiers.

2) Fitness convergence

The experiment results on fitness convergence, from all batches (2 configurations, 3 batches each, which has 2 tree structure types), are shown at table 2 and table 3, for an ideal data set and a practical data set, respectively. Average, standard deviation, as well as minimum and maximum of both # of generations and fitness (in the form of accuracy) are used to give a statistical performance of the method based on each batch.

Tree separation represents how the Valence and Arousal subtrees were evolved. *Combined* tree evolved both subtrees simultaneously, thus enabling crossover operations from Valence subtree with Arousal subtree, vice versa. *Separate* tree evolved both subtrees separately, thus crossover operations between them was not possible.

TABLE II. TRAINING CONVERGENCE RESULT USING IDEAL DATA SET

Voting Configuration				
Category	*Combined Tree*		*Separate Tree*	
	# Gen	Fitness %	# Gen	Fitness %
Worst	224	81.91	296	83.07
Best	51	95.59	84	97.79
Average	127.83	89.94	164.1	91.15
Stdev	40.64	3.77	32.74	3.64
Credibility Evolution Configuration				
Category	*Combined Tree*		*Separate Tree*	
	# Gen	Fitness %	# Gen	Fitness%
Worst	218	87.71	265	82.64
Best	70	99.97	75	99.14
Average	115.93	93.01	153.13	93.09
Stdev	41.46	3.44	35.08	4.97

TABLE III. TRAINING CONVERGENCE RESULT USING PRACTICAL DATA SET

Voting Configuration				
Category	*Combined Tree*		*Separate Tree*	
	# Gen	Fitness %	# Gen	Fitness %
Worst	302	85	226	82.42
Best	55	95.91	73	99.14
Average	135.27	91.02	128.97	92.63
Stdev	43.73	2.86	20.81	4.17
Credibility Evolution Configuration				
Category	*Combined Tree*		*Separate Tree*	
	# Gen	Fitness %	# Gen	Fitness %
Worst	143	84.55	234	82.16
Best	54	97.43	70	99.14
Average	92	91.27	128.4	92.4
Stdev	21.89	5.18	23.27	5.18

Current result showed that evolving Valence and Arousal subtrees separately gave a better result, but no improvement on the number of generations needed to reach termination. This is a similar result from our previous experiments.

By comparing the results between using *Voting* and *Credibility*, it can be seen that using *Credibility* gives a better result compared to using *Voting*.

3) Accuracy

Results regarding accuracy on *Test* data set is best described on table 4. The data shown are based on the statistical result (worst, best, average, and standard deviation) from each batch. As a reminder, a single batch consisted of several sessions, and each session consisted of several XGP runs, with each run consisted of several generations until termination.

Table 4 shows that using *Combined* tree and *Voting* gives better result, compared to using *Combined* tree and *Credibility*. Meanwhile, using a *Separate* tree and *Credibility* gives better result, compared to using *Separate* tree and *Voting*.

A simple observation on the table might give impressions that using *Combined* tree and *Voting* would be better in terms of accuracy, however there are several factors that should be considered as well. The programs evolved for the *Voting* and *Credibility* are different one another, and on some cases the convergence is better on the evolution of programs for the *Voting*.

Regardless, it is safe to conclude that despite the significant reduction on time needed to evolve the final classifiers, implementation of *Credibility* still has not yielded a better accuracy on Test set, compared with implementation of *Voting*. The possible cause is the lack of evolved programs variety, as our research so far only evolved several batches of 10 programs for *Voting* and 5 programs for *Credibility*.

TABLE IV. ACCURACY RESULTS ON TEST DATA SET

Ideal Data Set				
Category (%)	*Normal Voting*		*Credibility Evolution*	
	Combi	Separate	Combi	Separate
Worst	77.27	79.55	68.18	77.27
Best	90.91	81.82	79.55	81.82
Average	82.58	80.30	75.76	80.30
Stdev	7.31	1.31	6.56	2.62
Practical Data Set				
Category (%)	*Normal Voting*		*Credibility Evolution*	
	Combi	Separate	Combi	Separate
Worst	65.96	68.09	63.83	53.19
Best	76.60	74.47	68.09	74.47
Average	70.21	71.63	65.96	63.83
Stdev	5.63	3.25	2.13	10.64

IV. CONCLUSIONS

Using genetic algorithm to evolve the credibility of classifiers automatically gives a clear advantage in terms of computational performance, in the form of less time needed for evolution. Further, this method is capable of automated

evolution, as the classifiers are not selected by human. This method offers a possibility in completing a fully autonomous evolving intelligent agent.

The results acquired are not satisfactory enough, but it might be related to the insufficient number of data for training. Increasing training data might improve the result, as well as increasing the number of classifiers for voting and credibility evolutions.

For future works, another approach that can be done is to improve the training is by analyzing its contents and checking which terminal is dominant and which terminal is not used at all. By removing the terminals that are not used, the probability to converge better and faster also improves.

Another important improvement would be finding Pareto efficiency on the number of programs to be evolved and selected, in order to give a better improvement on time needed to evolve final classifiers, and also a better improvement on the accuracy.

REFERENCES

[1] R.W. Picard, "Affective computing." MIT Media Laboratory Perceptual Computing Section technical report no. 321.

[2] R.W. Picard, "Affective computing: From laughter to IEEE," IEEE Transactions on Affective Computing, Vol.1 No. 1, 2010.

[3] K. Takahashi, "Remarks on computational emotion recognition and expression," 6th international symposium on image and signal processing and analysis, 2009.

[4] P. Ekman, "Emotions revealed," Times books. 2003.

[5] P. Ekman, Friesen WV, "Facial action coding system." Consulting Psychologist Press, 1977.

[6] A. Pentland, "Honest signals," MIT press, 2008.

[7] J.A. Russel, "A circumplex model of affect," Journal of Personality and Social Psychology. 1980;39:1161–1178, 1980

[8] R. Plutchik, The nature of emotions. American Scientist, July 2001.

[9] R. Yusuf, S. Wang, I. Tanev, K. Shimohara, "Designing evolving computer agent capable of emotion recognition and expression," AAAI spring symposium. Palo Alto, California, 2014.

[10] R. Yusuf, I. Tanev, K. Shimohara, "Evolving emotion recognition module for intelligent agent," 18th Asia Pacific symposium on intelligent and evolutionary systems, Singapore, November 10-12, 2014, p215-226.

[11] R. Yusuf, D. G. Sharma, I. Tanev, K. Shimohara, " Evolving emotion recognition module of intelligent agent based on facial expression and gestures," 20th International symposium on artificial life and robotics, Japan, 2015., in press.

[12] J. Ahlberg, "CANDIDE 3 – an updated parameterized face," Report no LiTH-ISY-R-2326, Dept. of Electrical Engineering, Linköping University, Sweden, 2001.

[13] I. Tanev, K. Shimohara, "XML-based genetic programming framework: Design philosophy, implementation, and applications," Artificial-life and robotics, vol.15, no.4, December, pp.376-380, 2010

[14] I. Tanev, K. Shimohara, "XGP: XML-based genetic programming framework," Proceedings of the 34th Symposium of the Society of Instrument and Control Engineers (SICE) on Intelligent Systems, pp.183-188, March 15-16, 2007, Japan.

[15] B.W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," Biochimica et Biophysica Acta (BBA) – Protein Structure 405 (2), pp.442-451, 1975.