Journal of Computers Vol. 30 No. 2, 2019, pp. 27-40
doi:10.3966/199115992019043002003

EEG-based Motor Imagery Classification Using Novel Adaptive Threshold Feature Extraction and String Grammar Fuzzy K-Nearest Neighbor Classification

Payungsak Kasemsumran[1], Ekkarat Boonchieng[2*]

[1] Department of Computer Science, Thammasat University University, Bangkok and Lampang Campus, Thailand
payungsak2517@gmail.com

[2] Department of Computer Science, Center of Excellence in Community Health Informatics,
Chiang Mai University, Chiang Mai, Thailand
ekkarat.boonchieng@gmail.com

Received 15 December 2018; Revised 15 February 2019; Accepted 15 February 2019

Abstract. There has been great interest with the Motor Imagery (MI)–based brain–computer interface (BCI), recently. We developed the Novel Adaptive threshold for feature extraction. A string grammar fuzzy K-nearest neighbor is developed by incorporating 2 types of membership value into string grammar's K-nearest neighbor. The new algorithm used for the disabled or the patients who are physically unable to move with used to distinguish the order. In this experiment we check the decision to lift the arm. We apply these two-string grammar fuzzy K-nearest neighbors in the brain classification system. The system provides 88.14% in our dataset.

Keywords: brain classification, brain computer interface, EEG-based motor imagery classification, feature extraction, K-nearest neighbor, Novel Adaptive threshold, string grammar fuzzy K-nearest neighbor, string grammar

1 Introduction

One of the most attractive research topics for many generations of Scientists, Engineers, Practitioners, and Physicians is Motor imagery (MI)–based brain–computer interface (BCI). It is popular due to its profound importance in healthcare and in our future lives. The Adaptive threshold is the best algorithm and is suitable for many cases of segmentation [1-3]. A fixed threshold is not suitable in many cases. However, for some applications including a classification area, a large set of complex patterns, and structural information may be prefered over quantitative information. A signal pattern can be described by a sentence, e.g., a string or a tree or a graph, in a language [4-7]. There are several works in the Motor imagery (MI)–based brain–computer interface (BCI) system [8-9] because it will be the future in attractive applications and biomedical and health informatics field. Although, Motor imagery (MI)–based brain–computer interface (BCI) systems are up to standard, i.e., almost more than 64% classification rate.

However, for a Motor imagery (MI)–based brain–computer interface (BCI) system, it is not represented as structural information. String grammar can represent this structural information.

In this paper, we developed a Novel Adaptive threshold for feature extraction and string grammar fuzzy K-nearest neighbors (sgFKNN) based on two membership functions [10-11]. The feature extraction of the signal was developed by a Novel Adaptive threshold. Then, the brain signal information was extracted to form a string of symbols and then we applied our sgFKNN to classify each signal.

* Corresponding Author

2 Methodology

2.1 Dataset

The data set included the imaginative brain signals of a person who thought about raising their left arm and right arm. The descriptions of these data sets are described as follows:

We take the Electroencephalography (EEG) through 64 points on the head. The database contains the signals of 49 people, each of 6 blocks and release 3 minutes between blocks. For each individual, there are 100 trials each of 5 seconds, 64 channels EEG 2560 point-length of signal (5s), 100 trials (samples). Most of the signals were taken in two different sessions. First, they imagined raising their left arm and their right arm and were grouped into 2 classes with 9800 signals in each class. The total of the signals was 19600 for the 2 classes (the first is people thinking about raising their left arms and the second is people thinking about raising their right arms).

Fig. 1 shows the 64 points of the Electroencephalography (EEG) reading. After that, we have the signals that were taken during two different sessions observing the thinking about raising the left arm and right arm. The signals are shown in the time domain.

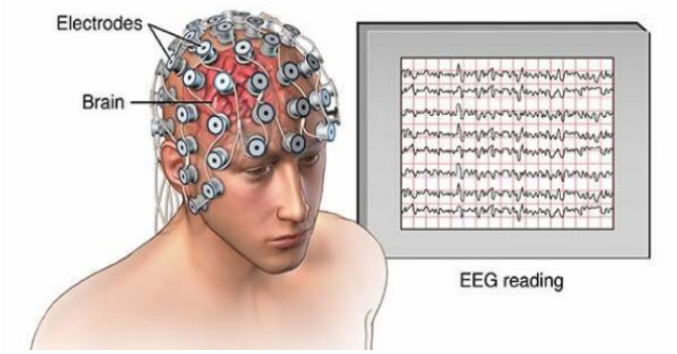

Fig. 1. EEG reading

Fig. 2 shows the signals of 49 people, each with 64 channels or points, there are 100 trials each for 5 seconds, 100 trials (samples).

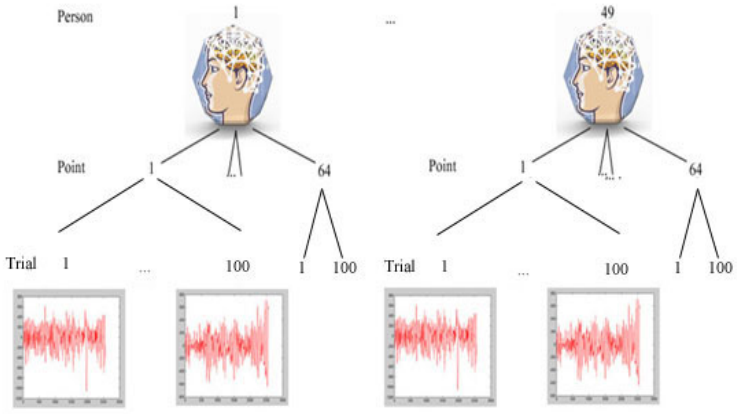

Fig. 2. Signal of raising left arm on each trials

Fig. 3 shows the example of raise left arm on 49 people, each with 64 channels or points.

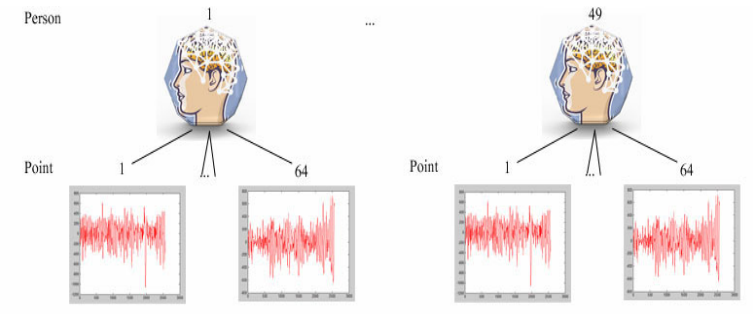

Fig. 3. Example of signal of raise left arm

Fig. 4 portrays the example of raising the right arm on 49 people, each with 64 channels or points.

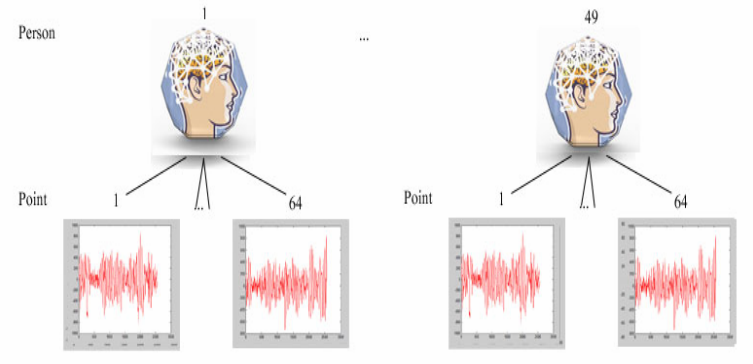

Fig. 4. Example of signals of raise right arm

2.2 Fast Fourier Transforms

The Fast Fourier Transforms (FFT) [12] is an algorithm that makes the computation of the Discrete Fourier Transform (DFT) possible in a time series quicker than other algorithms that are available. The DFT techngique is important due to the capabilities of computing such a fast algorithim.

$FFT\left(\overline{X}\right)$ is the discrete Fourier transform (DFT) of vector $\overline{X}$. For matrices, the FFT operation is applied to each column. For N-D arrays, the FFT operation operates on the first non-singleton dimension.

$FFT(\overline{X},N)$ is the N-point FFT, padded with zeros if $\overline{X}$ has less than N points and truncated if it has more.

$FFT(\overline{X},[],DIM)$ or $FFT(\overline{X},N,DIM)$ applies the FFT operation across the dimension DIM.

For length N input vector x, the DFT is a length N vector $\overline{X}$, with elements N.

$$\overline{X}(k)=\sum_{k=1}^{N}(x(n))\times\exp(-j\times 2\times \text{pi}\times(k-1)\times(n-1)/N) \tag{1}$$

The inverse DFT (computed by IFFT) is given by N.

$$x(n)=(\frac{1}{N})\sum_{n=1}^{N}\overline{X}(k)\times\exp(j\times 2\times pi\times(k-1)\times(n-1)/\text{N}) \tag{2}$$

Then, we have n-point FFT signals. After that, the signal of corresponding labeling 1 to corresponding labeling Z as:

$$S_{p1}=\begin{Bmatrix} sp1_1^1,...,sp1_{N_{T1}}^1,sp1_1^1,...,sp1_{N_{T2}}^1,sp1_1^1,...,sp1_{N_{TM}}^1,..., \\ sp1_1^C,...,sp1_{N_{T1}}^C,sp1_1^C,...,sp1_{N_{T2}}^C,sp1_1^C,...,sp1_{N_{TM}}^C \end{Bmatrix},$$

$$S_{p2}=\left\{\begin{matrix} sp2_1^1,...,sp2_{N_{T1}}^1,sp2_1^1,...,sp2_{N_{T2}}^1,sp2_1^1,...,sp2_{N_{TM}}^1,..., \\ sp2_1^C,...,sp2_{N_{T1}}^C,sp2_1^C,...,sp2_{N_{T2}}^C,sp2_1^C,...,sp2_{N_{TM}}^C \end{matrix}\right\},$$

$$\vdots$$

$$S_{pZ}=\left\{\begin{matrix} spZ_1^1,...,spZ_{N_{T1}}^1,spZ_1^1,...,spZ_{N_{T2}}^1,spZ_1^1,...,spZ_{N_{TM}}^1,..., \\ spZ_1^C,...,spZ_{N_{T1}}^C,spZ_1^C,...,spZ_{N_{T2}}^C,spZ_1^C,...,spZ_{N_{TM}}^C \end{matrix}\right\}$$

where S_{p1} is the signal of corresponding labeling 1

S_{pZ} is the signal of corresponding labeling Z

$sp1_1^C,...,sp1_{N_{TM}}^C$ are the signals on corresponding labeling1 in class C

N is the number of person

TM is trial number M

and ($s_{iT}^j \in \Sigma$ for $i=1,...,N_j, j=1,...,C,$ and $T=1,...,TM$).

Which we have the number of corresponding labeling as shown below.

Fig. 5 shows the EEG channel configuration-numbering (left) and the corresponding labeling (right) EEG data that were collected using 64 Ag/AgCl active electrodes. As shown in Fig. 1, a 64-channel montage based on the international 10-10 system was used to record the EEG signals with 512 Hz sampling rates. The EEG device used in this experiment was the Bio semi Active Two system [13].

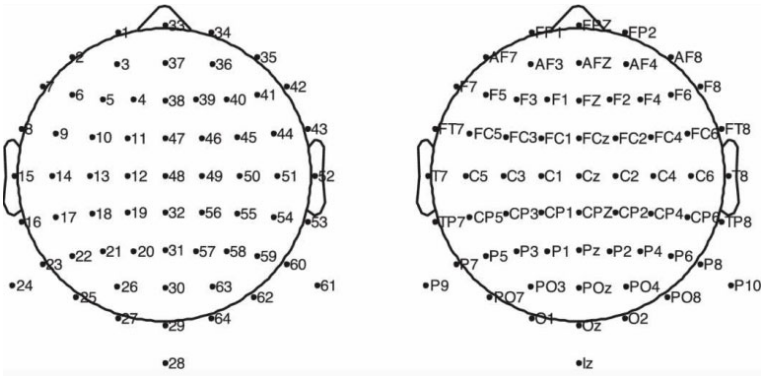

Fig. 5. EEG channel configuration-numbering (left) and corresponding labeling (right)

Fig. 6 shows the trained spatial filter for left/right hand MI. Electroencephalogram (EEG)-based brain-computer interfaces (BCIs) often use spatial filters to improve signal-to-noise ratio of task-related EEG activities. To obtain robust spatial filters, large amounts of labeled data, which are often expensive and labor-intensive to obtain, need to be collected in a training procedure before online BCI control [14-15].

The data set included the imaginative brain signals of a person who thought about raising their left arm and right arm. The descriptions of these data sets are described as follows:

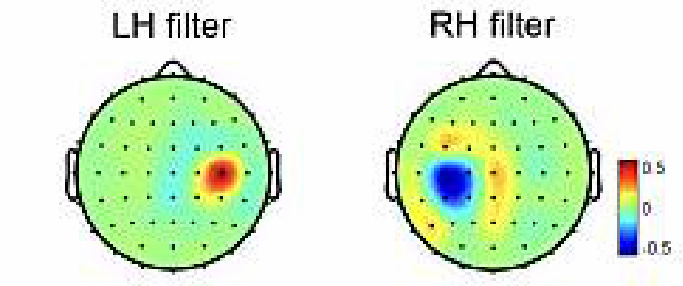

Fig. 6. Trained spatial filter for left/right hand MI

2.3 Novel Adaptive Thresholding

Fig. 7 and Fig. 8 found that the signals of the left arm on one person also had the same point and same trial as the left arm of a different person. They have similar signals however their frequencies are different. We have created a solution to this problem using "Novel Adaptive Threshold."

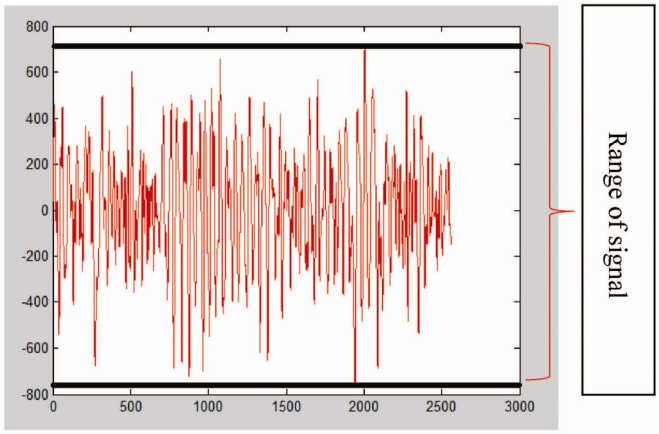

Fig. 7. The signal of raising left arm on 2nd person

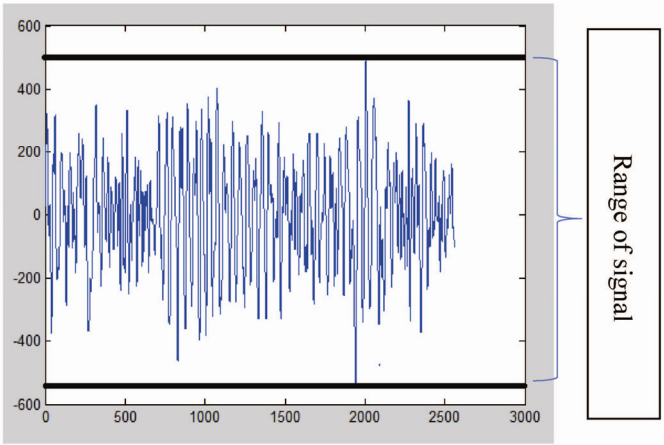

Fig. 8. The signal of raising left arm on 43rd person

We choose corresponding labeling, channel C3 and C4 for our experiment. They have the highest differential and clearest signal. We found it interesting that the signals have a large range in amplitude, however we do not care about the range of amplitude. For example, if the two signals which have different levels of amplitude have the same signal pattern, then the result will be the same class. e used the classic mathematic name "Class Interval". The Class Interval is the range of the data for each group. The first thing you need to decide is how many groups, classes, or levels, of data you want (we will call the number of levels because they are the not same with the data class in the classification). First, you would decide on a number of groups or levels for the data. Next, you would find the difference between the heaviest and lightest weights, which you may have already seen as an equation:

$$roundup(ci_{S_F}) = \frac{\max_{S_F} - \min_{S_F}}{\text{number of levels}} \tag{3}$$

where

S is the total signals for all S.

F are the number of signals for number 1 to 19800 (number of all signals).

If we already have the values of ci, then we always want our class interval to be a whole number, so in this case, it makes sense to use a class interval is roundup (ci_{S_F}).

2.4 String Generation Process

We used the difference of class intervals (ci_{S_F}) in each of the signals, but used the same number of levels. hus, we implemented an adaptive threshold method in each signal for the string generation.

One of the string sequences consists of strings in each of the levels for one signal. hen, the number of strings in one string sequence is equal to the number of levels.

2.5 String Grammar

Phrase structure grammar was originally introduced by Chomsky [16].

A phrase structure grammar G is a 4-tuple

$$G = (VN, VT, P, S) \quad \textbf{(4)}$$

where

VN finite set of nonterminal symbols
VT finite set of terminal symbols,
S start symbol,
P finite set of productions or rewrite rules of the form $\alpha \to \beta, \beta \in V, \alpha = \lambda$;
V* is the set of all finite length strings of symbols from V, including, the null string.
V (alphabet) is a finite (non-empty set of symbols)
λ is empty string

Let G = (VN, VT, P, S) be a grammar.

If every production in P is of the form

A→ aB, or
A →a,
A, B ∈ VN,
a ∈ VT,

then the grammar G is the finite-state or the regular grammar.

Syntactic pattern recognition has been described by phrase structure grammar. [17]. Each pattern is represented by a string of primitives corresponding to a sentence in a language (tree or graph in high dimensional grammars). All strings which belong to the same class are generated by one grammar.

2.6 String Grammar Fuzzy K-Nearest Neighbor (sgFKNN)

We have the training data set, i.e., strings of signals in all classes, is

$$\mathbf{X} = \left\{ \mathbf{x}_1^1, ..., \mathbf{x}_{N_1}^1, \mathbf{x}_1^2, ..., \mathbf{x}_{N_2}^2, ..., \mathbf{x}_1^C, ..., \mathbf{x}_{N_C}^C, \right\}$$

where $\mathbf{x}_i^j$ is a training string of signal *i* in class *j*, and N_j is the number of train strings of signals in class *j*. Each string ($\mathbf{x}_i^j$) is a sequence of symbols (primitives). For example, $\mathbf{x}_i^j = \left(x_{i1}^j x_{i2}^j ... x_{il}^j \right)$, a string with length *l*, where each x_{ir}^j is a member of a set Σ of defined symbols or primitives

$$\left(x_{ir}^j \in \Sigma \text{ for } i = 1, ..., N_j, j = 1, ..., C, \text{ and } r = 1, ..., l \right).$$

For a test string of signal x, the Levenshtein distances [18] between strings x and $\mathbf{x}_i^j$, $Lev\left(\mathbf{x}, \mathbf{x}_i^j\right)$, for $1 \le j \le C$ and $1 \le i \le N_j$ are calculated. Then K-closest strings of signals are identified. The membership value in [19] is modified to cope with the Levenshtein distance and used as a membership value of string **x** in class *l* as following:

$$u_l(\mathbf{x}) = \frac{\sum_{j=1}^{K} u_{ij}\left[\frac{\exp\left(-m\sqrt{C}Lev\left(\mathbf{x}-\mathbf{x}_j^q\right)\right)}{\beta}\right]}{\sum_{j=1}^{K}\left[\frac{\exp\left(-m\sqrt{C}Lev\left(\mathbf{x}-\mathbf{x}_j^q\right)\right)}{\beta}\right]} \tag{5}$$

Where u_{ij} is the membership value of the training string of image $\mathbf{x}_j^q$ in class i, m and C are the fuzzifier and the number of subjects in the training data set, respectively. Also, β is modified from [19] and calculated as:

$$\beta = \frac{\sum_{q=1}^{C}\sum_{j=1}^{N_q} Lev\left(\mathbf{x}_j^q, \mathbf{x}_{med}\right)}{\sum_{q=1}^{C} N_q} \tag{6}$$

Where the median string $\mathbf{x}_{\text{med}}$ in a set of strings X can be calculated as [20-21]

$$\mathbf{x}_{med} = \arg\min_{j\in\mathbf{X}} \sum_{k=1}^{N_1+N_2+\ldots+N_C} Lev(\mathbf{x}_j, \mathbf{x}_k). \tag{7}$$

Then, the decision is as follows:

$$\mathbf{x} \text{ is assigned to class } i \text{ if } u_i(\mathbf{x}) > u_j(\mathbf{x}) \text{ for } j \neq i. \tag{8}$$

In our experiment, since we know that the training string of image $\mathbf{x}_j^q$ represents which class, we set $u_{jq} = 1$ for $\mathbf{x}_j^q$ in class q and 0 for all the other classes. Also, we set $m = 2$ in our experiment. For the initialization of membership or assigned membership, we will explain in the next section.

To make a comparison, we implemented our sgFKNN with the membership value modified from [11] as

$$u_i(\mathbf{x}) = \frac{\sum_{j=1}^{K} u_{ij}\left(1/Lev\left(\mathbf{x}-\mathbf{x}_j^q\right)\right)^{2/m-1}}{\sum_{j=1}^{K}\left(1/Lev\left(\mathbf{x}-\mathbf{x}_j^q\right)\right)^{2/m-1}}. \tag{9}$$

The parameters m and the membership uiq are set the same way as in equation 5. To make it simple, in the experiment, we call the sgFKNN with membership values computed as in equations 5 and 9 as the sgFKNN1 and sgFKNN2, respectively.

2.7 Initialization of Membership

The membership values have several properties as follows:

(1) It should be 1.0 if the vector is equal to the mean of its class.

(2) It should be 0.5 if the vector is equal to the mean of the other class.

(3) It should be near 0.5 if the vector is equidistant from the two means.

(4) It should never be less than 0.5.

(5) The membership value should approach 1.0 exponentially as a vector gets closer to its mean and farther from the other mean

(6) It should depend on relative distances from the means of the classes rather than absolute distances.

The membership initialization [11] or assigned memberships of $\mathbf{x}$ are influenced by the inverse of the distances from the nearest neighbors and their class memberships. The inverse distance serves to weigh a vector' membership more if it is closer and less if it is farther from the vector under consideration. The

labeled samples can be assigned class memberships in several ways. First, they can be given complete memberships in their known class and non-membership in all other classes. Other alternatives are to assign the sample's membership based on distance from their class mean or based on the distance from labeled samples of their own class and those of the other class or classes, and then to use the resulting memberships in the classifier. Both of these techniques have been used in this study and the results are reported. It is noted that in [22] an alternate scheme for assigning initial memberships based on a learning scheme was considered.

The variable m determines how heavily the distance is weighted when calculating each neighbor's contribution to the membership value. If m is two, then the contribution of each neighboring point is weighed by the reciprocal of its distance from the point being classified. As m increases, the neighbors are more evenly weighted, and their relative distances from the point being classified have less effect. As m approaches one, the closer neighbors are weighted far more heavily than those farther away, which have the effect of reducing the number of points that contribute to the membership value of the point being classified. We used $m=2$, but note that almost equal error rates have been obtained on these data over a wide range of values of m.

Many other researches have explained the types of labeling techniques used for the fuzzy classifier. For example, in [11] there are three different techniques of membership assignment for the labeled data that are considered.

1st method. We call this "crisp labeling". The first method assigns the membership values based on each labeled sample complete membership in its known class and zero membership in all other classes.

2nd method. The second method assigns membership based on the procedure presented in [23]. This technique works only on two class data sets. The procedure assigns a sample membership in its known class based on its distance from the mean of the labeled sample class. These memberships range from one to one half with an exponential rate of change between these limits. The sample's membership in the other class is assigned such that the sum of the memberships of the vector equals one. A more detailed explanation of this technique is given in [23].

3rd method. The third technique assigns memberships to the labeled samples according to a K-nearest neighbor rule. The K (not K of the classifier)-nearest neighbors to each sample x (say x in class i)are found, and then membership in each class is assigned according to the following equation:

$$u_j = \begin{cases} 0.51 + 0.49(n_j / K), & if\ j = i \\ 0.49(n_j / K), & if\ j \neq i \end{cases} \quad (\mathbf{10})$$

where

n_j is the number of the neighbors found which belong to the j class.

This method attempts to "fuzzify" the memberships of the labeled samples, which are in the class regions intersecting in the sample space, and leaves the samples that are well away from this area with complete membership in the known class. Since this method, an unknown sample lying in this intersecting region will be influenced to a lesser extent by the labeled samples that are in the "fuzzy" area of the class boundary.

The second technique assigns membership based on the procedure presented in [23]. The membership in each class is assigned according to the following equation:

For $\overrightarrow{x_k}$ in class 1:

$$u_{1k} = 0.5 + \frac{\exp(f(d_2 - d_1)/d) - \exp(-f)}{2(\exp(f) - \exp(-f))} \quad (\mathbf{11})$$

and

$$u_{2k} = 1 - u_{1k} \quad (\mathbf{12})$$

For $\overline{x_k}$ in class 2:

$$u_{1k} = 1 - u_{2k} \tag{13}$$

and

$$u_{2k} = 0.5 + \frac{\exp(f(d_1 - d_2)/d) - \exp(-f)}{2(\exp(f) - \exp(-f))} \tag{14}$$

where

d_1 and d_2 are the distances from the testing sample vector to the center of class 1 and class 2, respectively.

d is the distance between the two centers of each class.

f is a parameter used for controlling the rate at which memberships decrease toward 0.5, normally $f > 0$.

2.8 Our Algorithm

Step 1. Receive the data from 49 people. Most of the signals were taken in two different sessions to record the thoughts of raising the left arm and the right arm, grouped into 2 classes with 9,800 signals in each class then the total of signals are 19,600 signals.

Step 2. Initialize the membership values of each signal.

Step 3. Compute s_{iT}^{j} using Eq. (1) and (2).

Step 4. Compute ci_{S_F} using Eq. (3).

Step 5. String generation.

Step 6. Compute $u_l(\mathbf{x})$ using Eq. (5).

Step 7. Compute the accuracy rate

In step 6, a fuzzy K-nearest neighbor algorithm is as follows:

```
BEGIN
  Input x, of unknown classification.
  Set K, 1 ≤ K ≤ n.
  Initialize i = 1
  DO UNTIL (K-nearest neighbors of x⃗ are found)
        Compute distance from x⃗ to x⃗_i
    IF (i ≤ K) THEN
        Include x⃗_i in the set of K-nearest neighbors
    ELSE IF (x⃗_i closer to x⃗ than any previous nearest neighbor) THEN
        Delete the farthest of the K-nearest neighbors
        Include x⃗_i in the set of K-nearest neighbors.
    END IF
  END DO UNTIL
  Initialize i = 1.
  FOR i = 1 to c
    Compute u_i(x⃗) using (5).
    Increment i.
  END
```

3 Experimental Results

We implemented the FFT and novel adaptive threshold to generate strings from the signals. Next, we implemented the new string grammar fuzzy K-nearest neighbor (sgFKNN1) from equation (5) and string grammar fuzzy K-nearest neighbor (sgFKNN2) from equation (9) on the data set for classification.

After that, we implemented the leave-one-out scheme and used the same setting for all of the data sets. We also set K-nearest neighbor to the number of signals of each class minus 1. For example, for left hands, there are 9800 signals for class, hence K-nearest neighbor would be set to 9799 (N-signals per class minus 1). The same scheme was utilized in all of the data sets. The recognition rate from the sgFKNN1 and sgFKNN2 on data set, are shown in Table 1. We also decided to increase the performance of the data set by increasing the number of sub-signals in the string generation process; hence, there were 10 sub-signals. We also directly compared the sgFKNN1, sgFKNN2 results as shown in Table 1.

Table 1. The recognition rate from sgFKNN1 and sgFKNN2 on the data sets with string length of 10

K	sgFKNN1	sgFKNN2
1	88.14%	68.27%
5	87.71%	68.24%
9	87.71%	68.24%
8999	70.15%	60.99%
N signal per class-1	70.15%	60.99%

Table 1 shows the results from our algorithm from sgFKNN1 and sgFKNN2 on the EEG dataset. We can see that the system provides the high recognition rates as a small K. Although, the recognition rates from our system are not close to 100%, due to the fact that the physical signals make more noise. However, the recognition rates from our system are close to 90%, which is not bad. Never the less, we have a direct comparison with fuzzy K-nearest neighbor [11] through which we have the highest accuracy rates in every K value. Another reason for the misclassification might be due to the transformation of the strings used in the calculation of the Levenshtein distance. This might create a small distance between strings that are actually farther apart in the normal sense.

Fig. 9 shows the membership values on class 2 which are correctly classified and values that are not close to between class 1 and class 2.

		Membership values in each class		RMSE
	Signal No.	1	2	(Class 1,1)
Number of signal on class 1	4990	0.6154	0.3846	0.3846
	4991	0.8078	0.1922	0.1922
	4992	0.8286	0.1714	0.1714
	4993	0.8286	0.1714	0.1714
	4994	0.8286	0.1714	0.1714
	4995	0.8286	0.1714	0.1714
	4996	0.8286	0.1714	0.1714
	4997	0.8286	0.1714	0.1714
	4998	0.7900	0.2100	0.2100
	4999	0.7900	0.2100	0.2100
	5000	0.7900	0.2100	0.2100
	5001	0.5732	0.4268	0.4268
	5002	0.8286	0.1714	0.1714
	5003	0.8078	0.1922	0.1922
	5004	0.8078	0.1922	0.1922

Fig. 9. The membership values on class 1 which they are correct classification

On the other hand, Fig. 10 shows that the membership values on class 2 where they are misclassification but values are close to between class 1 and class 2.

		Membership values in each class		RMSE
	Signal No.	1	2	(Class 2,1)
Number of signal on class 2	12940	0.5405	0.4595	0.5405
	12941	0.5062	0.4938	0.5062
	12942	0.5056	0.4944	0.5056
	12943	0.5374	0.4626	0.5374
	12944	0.5336	0.4664	0.5336
	12945	0.5336	0.4664	0.5336
	12946	0.5336	0.4664	0.5336
	12947	0.5336	0.4664	0.5336
	12948	0.5336	0.4664	0.5336
	12949	0.5336	0.4664	0.5336
	12950	0.5336	0.4664	0.5336
	12951	0.5732	0.4268	0.5732
	12952	0.5732	0.4268	0.5732
	12953	0.5732	0.4268	0.5732
	12954	0.5732	0.4268	0.5732

Fig. 10. The membership values on class 2 where they are misclassification

Never the less, we used "The Root Means Square Error" (RMSE) (also called the root mean square deviation, RMSD) which is a frequently used measure of the difference between values predicted or membership value (value of 1) by a model and the values actually observed from the environment or the membership value in class that is being modelled. These individual differences are also called residuals, and the RMSE serves to aggregate them into a single measure of predictive power.

The RMSE of a model prediction with respect to the estimated variable X_{model} is defined as the square root of the mean squared error:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n}} \quad (15)$$

Where X_{obs} is observed values and X_{model} is modelled values at time/place i.

The calculated RMSE values will show the error of the classification in each test signals.

Figure 11 and Figure 12 show strings on class 1 and class 2 which they are correct classification, respectively. One the other hand, one of the problems is that the Levenshtein distance is based on the transformation of the strings that may not be appropriate for computing the distance since there may be two strings that are close from the view of this distance, but actually they are far apart in the normal sense. This might happen from two different faces with two similar strings. However, we suspect that increasing the number of sub signals (symbols) might help in this case.

4990	[10,1,10,10,10,1,10,10,10,1]
4991	[10,2,1,1,1,1,1,1,1,2]
4992	[10,1,1,1,1,1,1,1,1,1]
4993	[10,1,1,1,1,1,1,1,1,1]
4994	[10,1,1,1,1,1,1,1,1,1]
4995	[10,1,1,1,1,1,1,1,1,1]
4996	[10,1,1,1,1,1,1,1,1,1]
4997	[10,1,1,1,1,1,1,1,1,1]
4998	[10,10,2,1,1,1,1,2,10]
4999	[10,10,2,1,1,1,1,2,10]
5000	[10,10,2,1,1,1,1,2,10]
5001	[10,10,10,10,10,10,10,10,10,10]
5002	[10,1,1,1,1,1,1,1,1,1]
5003	[10,2,1,1,1,1,1,1,1,2]
5004	[10,2,1,1,1,1,1,1,1,2]

Fig. 11. Strings on class 1 which are correctly classified

12940	[1,10,1,10,2,1,2,10,1,10]
12941	[1,10,2,10,1,3,1,10,2,10]
12942	[1,10,2,10,1,4,1,10,2,10]
12943	[1,10,3,10,1,7,1,10,3,10]
12944	[10,10,10,10,1,10,1,10,10,10]
12945	[10,10,10,10,1,10,1,10,10,10]
12946	[10,10,10,10,1,10,1,10,10,10]
12947	[10,10,10,10,1,10,1,10,10,10]
12948	[10,10,10,10,1,10,1,10,10,10]
12949	[10,10,10,10,1,10,1,10,10,10]
12950	[10,10,10,10,1,10,1,10,10,10]
12951	[10,10,10,10,10,10,10,10,10,10]
12952	[10,10,10,10,10,10,10,10,10,10]
12953	[10,10,10,10,10,10,10,10,10,10]
12954	[10,10,10,10,10,10,10,10,10,10]

Fig. 12. Strings on class2 which they are correct classification

From all the results, we found out that this sgFKNN1 is better than sgFKNN2 methods in every K value. However, the sgFKNN1 still provides a high accuracy rate. This is because of the nature of the algorithm that is needed to be improved in the future.

Never the less, we want to know something about the accuracy rate when we aren't using the Fast Fourier Transform (FFT) from the methodology.

The Fast Fourier Transform (FFT) is the good method of calculating harmonics not one at a time, but as a group, using a special algorithm. The FFT requires much less processing power than a DFT for the same number of harmonic results. Fast Fourier Transforms are faster than Discrete Fourier Transforms. Check the example below to see how much faster they are.

For example, the waveform of 1024 samples, N, it takes N^2 computations to calculate the harmonics, while for a FFT it takes $N \log_2 N$ computations. So, for the DFT it takes 1048576 computations and for the FFT it takes 10240 computations. The FFT is over 100 times faster. However, the number of computations given is for calculating 1024 harmonics from 1024 samples.

The Fast Fourier Transform (FFT) is a better and fasterer method compared to the Discrete Fourier Transforms (DFT) however The Fast Fourier Transform still needs to take some time to work.

We are interested in the computation time or time complexity and the accuracy rate. If we reduce some methods in our algorithm, then the computation time as well will be decreased. The accuracy rate will increase or decrease.

As mentioned before, we implemented the same methodology but by not using the Fast Fourier Transforms (FFT) from our algorithm which the new algorithm which follows these steps below:

Step 1. Acquisition of the data from 49 persons. Most of the signals were taken in two different sessions to take into raise left arm and right arm imagines, grouped into 2 classes with 9800 signals in each class then the total of signals are 19600 signals.

Step 2. Initialize the membership values of each signals.

Step 3. Compute s_{iT}^{j} without Eq. (1) and (2) but using code for sampling signals.

Step 4. Compute ci_{S_F} using Eq. (3).

Step 5. String generation.

Step 6. Compute $u_l(\mathbf{x})$ using Eq. (5).

Step 7. Compute the accuracy rate

Table 2 shows the direct comparison of the results from sgFKNN1 (with FFT), sgFKNN1 (without FFT) and sgFKNN2 on the EEG dataset with string length of 10.

Table 2. The direct comparison of the recognition rate from sgFKNN1, sgFKNN1(new) and sgFKNN2 on the data sets with string length of 10

K	sgFKNN1 (with FFT)	sgFKNN1 (without FFT)	sgFKNN2
1	88.14%	78.21%	68.27%
5	87.71%	77.98%	68.24%
9	87.71%	77.98%	68.24%
8999	70.15%	65.57%	60.99%
N signal per class-1	70.15%	65.57%	60.99%

The sgFKNN1 (without FFT) is the same algorithm with sgFKNN1 but the sgFKNN1 (without) is not using the Fast Fourier Transforms (FFT).

During our experimentation we found that the accuracy rates of sgFKNN1 with Fast Fourier Transform (FFT) are better than sgFKNN1without Fast Fourier Transform (FFT) in every K value. In order to determine which of these two algorithms, depend on your needs. If you want the lowest of time processing, you must choose the sgFKNN1 without FFT. However, if you want a high accuracy rate, you must choose the sgFKNN1 with FFT.

Acknowledgements

The authors would like to thank the Computer Science Department, Faculty of Science, Thammasat University, Bangkok and Lampang Campus, Thailand and Center of Excellence in Community Health Informatics, Chiang Mai University, Kantary Terrace Building, 3rd floor, Nimman Rd., Suthep, Muang, Chiang Mai, Thailand, for valuing and foreseeing the upcoming of computing technology to serve the public.

References

[1] H.Y. Chan Francis, F.K. Lam, H. Zhu, Adaptive thresholding by variational method, IEEE Transections on Image Processing 7(3)(1998) 468-473.

[2] D. Agarwal, N. Kishor, A fuzzy inference-based fault detection scheme using adaptive thresholds for health monitoring of offshore wind-farms, IEEE Sensors Journal 14(11)(2014) 3851-3861.

[3] F.P.S. Luus, F. van den Bergh, B.T.J. Maharaj, Adaptive threshold-based shadow masking for across-date settlement classification of panchromatic quickbird signals, IEEE Geoscience and Remote Sensing Letters 11(6)(2014) 1153-1157.

[4] K.S. Fu, S.Y. Lu, A clustering procedure for syntactic patterns, IEEE Transactions on Systems Man and Cybernetics 7(10)(1977) 734-742.

[5] A. Juan, E. Vidal, On the use of normalized edit distances and an efficient k-NN search technique (k-AESA) for fast and accurate string classification, in: Proc. 15th International Conference on Pattern Recognition, 2000.

[6] J.C. Bezdek, J. Keller, R. Krishnapuram, N.R. Pal, Fuzzy Models and Algorithms for Pattern Recognition and Image Processing, Kluwer Academic, Norwell, 1999.

[7] B. Zhou, X. Wu, Z. Lv, L. Zhang, X. Guo, A fully automated trial selection method for optimization of motor imagery based brain-computer interface, PLoS One 11(9)(2016) e0162657. doi: 10.1371/journal.pone.0162657.

[8] Y. Zhang, G. Zhou, J. Jin, X. Wang, A. Cichocki, Optimizing spatial patterns with sparse filter bands for motor-imagery based brain–computer interface, Journal of Neuroscience Methods 255(2015) 85-91.

[9] K. Kitahara, Y. Hayashi, T. Kondo, S. Yano, Sound imagery contributes to foot MI-based BCI even through it does not influence on the sensorimotor rhythms, in: Proc. Asia-Pacific Signal and Information Processing Association Annual

Summit and Conference (APSIPA), 2016.

[10] S. Siuly, Y. Li, Improving the separability of motor imagery EEG signals using a cross correlation-based least square support vector machine for brain–computer interface, IEEE Transections on Neural Systems and Rehabilitation Engineering 20(4)(2012) 526-538.

[11] M.S. Yang, K.L. Wu, Unsupervised possibilistic clustering, Pattern Recognition 39(1)(2006) 5-21.

[12] J.M. Keller, M.R. Gray, J.A. Givens Jr., A fuzzy K-nearest neighbor algorithm, IEEE Transections on Systems Man and Cybernetics SMC-15(4)(1985) 580-585.

[13] W.T. Cochran, J.W. Cooley, D.L. Favin, H.D. Helms, R.A. Kaenel, W.W. Lang, G.C. Maling, D.E. Nelson, C.M. Rader, P.D. Welch, What is the fast fourier transform?, Proc. TH IEEE 55(10)(1967) 1664-1674.

[14] H. Cho, M. Ahn, S. Ahn, M. Kwon, S.C. Jun, EEG datasets for motor imagery brain–computer interface, Gigascience 6(7)(2017) 1-8. doi: 10.1093/gigascience/gix034.

[15] Y. Wang, Y.T. Wang, T.P. Jung, Translation of EEG spatial filters from resting to motor imagery using independent component analysis, PLoS One 7(2012) e37665. doi: 10.1371/journal.pone.0037665.

[16] A. Curtin, H. Ayaz, Y. Liu, P.A. Shewokis, B. Onaral, A P300-based EEG-BCI for spatial navigation control, in: Proc. AnnualInternational Conference of the IEEE Engineering in Medicine and Biology Society, 2012. doi: 10.1109/EMBC. 2012.6346805.

[17] N. Chomsky, Syntactic Structures, Mouton & Co., Paris, 1957.

[18] K.S. Fu, Syntactic Pattern Recognition and Applications, Springer Science+Business Media, Berlin, 1982.

[19] J.C. Bezdek, J. Keller, R. Krishnapuram, N.R. Pal, Fuzzy Models and Algorithms for Pattern Recognition and Image Processing, Springer Science & Business Media, Berlin, 1999.

[20] M.S. Yang, K.L. Wu, Unsupervised possibilistic clustering, Pattern Recognition 39(1)(2006) 5-21.

[21] T. Kohonen, Median strings, Pattern Recognition Letters 3(5)(1985) 309-313.

[22] A. Klomsae, S. Auephanwiriyakul, N. Theera-Umpon, A novel string grammar fuzzy C-medians, in: Proc. the 2015 IEEE International Conference on Fuzzy Systems, 2015.

[23] A. Jozwik, A learning scheme for a fuzzy k-NN rule, Pattern Recognition Letters 1(5-6)(1983) 287-289.