High Performance Classification of Two Imagery Tasks in the Cue-Based Brain Computer Interface

Omid Dehzangi, Mansoor Zolghadri Jahromi, and Shahram Taheri

School of Computer Engineering,
Nanyang Technological University, Nanyang Avenue, Singapore
`Omid0002@ntu.edu.sg`,
`{Zjahromi,Taheri}@cse.shirazu.ac.ir`

Abstract. Translation of human intentions into control signals for a computer, so called Brain-Computer Interface (BCI), has been a growing research field during the last years. In this way, classification of mental tasks is under investigation in the BCI society as a basic research. In this paper, a Weighted Distance Nearest Neighbor (WDNN) classifier is presented to improve the classification rate between the left and right imagery tasks in which a weight is assigned to each stored instance. The specified weight of each instance is then used for calculating the distance of a test pattern to that instance. We propose an iterative learning algorithm to specify the weights of training instances such that the error rate of the classifier on training data is minimized. ElectroEncephaloGram (EEG) signals are caught from four familiar subjects with the cue-based BCI. The proposed WDNN classifier is applied to the band power and fractal dimension features, which are extracted from EEG signals to classify mental tasks. Results show that our proposed method performs better in some subjects in comparison with the LDA and SVM, as well-known classifiers in the BCI field.

Keywords: Nearest Neighbor, Weighted distance, Brain-Computer Interface, EEG.

1 Introduction

Classification of mental imagery tasks is used to help amyotrophic lateral sclerosis (ALS) patients to enable them to communicate with their environment [1]. A bright view to the future of this research is to help ALS patients by enabling them to move their limbs with their thoughts. Limb movement can be done by Functional Electrical Stimulation (FES) [2], which is controlled by the BCI system. This interesting application is in its primary stages mainly due to low classification rate even between two imagery tasks in some subjects.

The research in the BCI field can be categorized into *synchronous* [1] and *asynchronous* [3] methods. Most articles focus on the synchronous BCI which is so called cue-based BCI. In this way, Boostani *et al.* [4] applied Adaboost classifier on the fractal dimension features (extracted from the EEG signals) and showed that this combination has a good prediction ability. In a comprehensive study, Boostani *et al.* [5] employed genetic algorithm on different features and used three different classifiers on the weighted features to show that choosing the band power and fractal dimension as features (by genetic weighting) can significantly improve the performance of cue-based BCI system. The Graz-BCI research group has employed discriminative features based on second order statistics such as band power [1], adaptive autoregressive coefficients [6], and wavelet coefficients [7] with well-known classifiers containing Fisher's Linear Discriminant Analysis (FLDA) [8], Finite Impulse Response Multi-Layer Perceptrons (FIRMLP) [9], Linear Vector Quantization (LVQ) [10], Hidden Markov Models (HMM) [1], and Distinction Sensitive Learning Vector Quantization (DSLVQ) [11] to improve the classification rate between the various movement in imagery tasks. Deriche *et al.* [12] selected the best feature combination among variance, AR coefficients, wavelet coefficients, and fractal dimension by modified mutual information method. They showed that a combination of the aforementioned features is more efficient than each of them individually.

As a simple but efficient supervised learning algorithm, the nearest neighbor classifier has been used successfully on pattern classification problems [13], [14]. However, this method fails to perform satisfactorily in cases that different classes are overlapped in some regions of feature space. Another problem is the noisy training instances that can degrade the performance of this classifier in the generalization phase.

The basic NN uses all training data in the generalization phase. It also considers all the stored instances with the same importance for classification, but the instances are different in being representative of their typical classes.

Recently, many improving techniques have been proposed and added to the nearest neighbor algorithm such as editing, condensing, learning, and weighting [15] for overcoming to its drawbacks. Moreover, there has been considerable research interest in learning mechanisms to locally adapt the distance metrics [16], [17]. Wang et al. [18], [19] have shown that by including a local weight and introducing a simple adaptive distance measure the performance of the NN improves significantly. In this paper a novel learning algorithm is presented which is used to assign a weight to each stored instance, which is then contributed in distance measure, with the goal of improvement in generalization ability of the basic NN. Our proposed learning method is used to adjust the weights of instances in the training set. The basic component of the learning algorithm is an optimization procedure that finds the best operating point of a classifier (i.e., resulting in minimal error rate of the classifier on train data). The proposed scheme achieves two desirable goals at the same time. The classification rate is improved by adjusting a weight for each instance and considering it while calculating distance measure. Our experiments show that the proposed WDNN algorithm can make a robust and accurate classifier system that improves the performance of the cue-based BCI.

The rest of this paper is organized as follows. In section 2, subjects and the method of data acquisition are described. In section3, features are illustrated. In section 4, the proposed WDNN and our proposed method of learning the weights of training instances are described. In section 5, the experimental results are presented and in section 6, conclusion is discussed.

2 Subjects and Data Acquisition

Four subjects (L1, O3, O8, and G8), familiar with the Graz-BCI, participated in this study. Subjects are ranged from 25 to 35 years old. Each subject sat in a armchair about 1.5 meters in front of the computer screen. Three bipolar EEG-channels were recorded from 6 Ag/AgCl electrodes placed 2.5 cm anterior and 2.5 cm posterior to the standardized positions C3, Cz and C4 (international 10-20 system). The EEG was filtered between 0.5 and 50 Hz and recorded with a sample frequency of 128 Hz.

The training in Graz-BCI paradigm is consisted of a repetitive process of triggered movement imagery trials. Each trial lasted 8 seconds and started with the presentation of a blank screen. A short acoustical warning tone was presented at second 2 and a fixation cross appeared in the middle of the screen. At the same time, the trigger was set from 0 to 1 for 500 milliseconds. From second 3 to second 7, the subjects performed left or right hand motor imagery according to an arrow (cue) on the screen. An arrow pointing either to the left or to the right indicated the imagination of a left hand or right hand movement. The order of appearance of the arrows was randomized and at second 7 the screen content was erased. The trial finished with the presentation of a randomly selected inter-trial period (up to 2 seconds) beginning at second 8. Figure 1. shows the timing scheme. Three sessions were recorded for each subject on 3 different days. Each session consisted of 3 runs with 40 trials each.

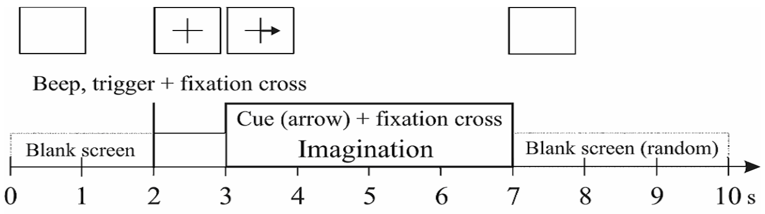


Fig. 1. Training paradigm

3 Feature Extraction

The goal of feature extraction is to find an informative representation of the data that simplifies the detection of brain patterns. The signal features should encode the commands sent by the user. Band power and fractal dimension features are used in this paper. These are briefly described in the following sections.

3.1 Band Power (BP)

The EEG contains different specific frequency bands, that is standard alpha (10-12Hz) and beta (16-24Hz) bands, which are particularly important in classifying different brain states, especially for discriminating imagery tasks. For this study, band power features were calculated by applying a Butterworth filter (order 5), squaring of the samples and then averaging of subsequent samples (1 s average with 250 ms overlap).

3.2 Fractal Dimension (FD)

BP and AAR features are based on the second order statistics of the signal and thus they describe the spectral information in the data. FD, however, captures nonlinear dynamics in the signal. Although all features here try to capture the underlying neurophysiological patterns in the signal, FD has a direct relationship with the entropy of the signal, which in turn is related to information content of the signal. FD is a measure of complexity of a signal. More fluctuation in the attractor shape is reflected by a higher value of FD. There are several methods to calculate the FD [20]. In this study we employed Higuchi's method [21], which is described as follows: Consider a signal containing N samples $\{x(1), x(2),\ldots,x(N)\}$. Construct k new time series x_m^k (embedded space) as:

$$x_m^k = \left\{ x(m), x(m+k), x(m+2k), \ldots, x\left(m + \left[\frac{N-m}{k}\right]k\right) \right\} \; for \quad m=1,\ 2, \ldots,\ k. \tag{1}$$

where m indicates the initial time value, and k represents the discrete time interval between points. For each of the k time series x_m^k, the length $L_m(k)$ is computed by:

$$L_m(k) = \frac{\sum_{i=1}^{\left[\frac{N-m}{k}\right]} \left| x(m+ik) - x(m+(i-1)k) \right| (N-1)}{\left[\frac{N-m}{k}\right]k} \tag{2}$$

where N is the total length of the data sequence x and $(N\text{-}1)/[(N\text{-}m)/k]k$, is a normalization factor. An average lengyth of every sub-sequence is computed as the mean of the k lengths $L_m(k)$. This procedure is repeated for the different values of k ($k = 1,2, \ldots,k_{max}$), that k_{max} varies for each k. There is no analytical formula for determining the value of k, therefore, it has to be found experimentally. An average length for each k is obtained which may be expressed as proportional to k^{-D}, where D is the signal's FD. In order to find the best value of k, from the log-log plot of $log(L(k))$ versus $log(1/k)$, one obtains the slope of the least-squares linear best fit. The FD of the signal, D, is then calculated as:

$$D = [\log L(k)] \,/\, \log(1/k) \tag{3}$$

4 Weighted Distance Nearest Neighbor (WDNN)

We briefly describe the NN rule to introduce the notation. For an M-class problem, assume that a set of training examples of the form $\{(X_i, C_i) \mid i = 1,\ldots, N\}$ is given. Where, X_i is a n-dimensional vector of attributes $X_i = [x_{i1}, x_{i2}, \ldots,x_{in}]^T$ and $C_i \in [1,2, \ldots,M]$ defines the corresponding class label. To identify the NN of a query pattern Q, a distance function has to be defined to measure the distance between two patterns. Euclidean distance has conventionally been used to measure the distance (i.e., dissimilarity) between two patterns X_i and X_j:

$$\mathrm{d}(X_i, X_j) = \sqrt{\sum_{k=1}^{n} \left(x_{ik} - x_{jk}\right)^2} \tag{4}$$

Assuming that each attribute of the problem is normalized to the interval [0,1], we can equivalently work with the following similarity measure (instead of Euclidean dissimilarity measure), which normalizes the similarity between two instances X_i and X_j to a real number in the interval [0,1]:

$$\mu(X_i, X_j) = 1 - \frac{\mathrm{d}(X_i, X_j)}{\sqrt{n}} \tag{5}$$

With basic NN rule, the query pattern Q is classified by the class most similar training pattern X_p in the training set. This can be formally stated as:

$$\mathrm{p} = \underset{1 \le i \le N}{\operatorname{argmax}} \left\{\mu(Q, X_i)\right\} \tag{6}$$

The NN rule assumes that all classifiers (i.e., stored instances) are equally reliable and uses equation (6) to find the NN of a query pattern. This paper is based on the idea that some of the stored instances are more reliable classifiers than others. We accomplish this by assigning a weight w_k to each instance X_k. The weights of the training instances are used in the test phase to find the NN of a query pattern:

$$\mathrm{p} = \underset{1 \le j \le N}{\operatorname{argmax}} \left\{w_j \times \mu(Q, X_j)\right\} \tag{7}$$

We refer to this classifier as WDNN. Alternatively, the scheme can be viewed as a form of adaptive distance measure for NN that allow the distance measure to vary as a function of instances in the training set. In the next section, we present an algorithm that finds the best operating point in 2-class problems. This algorithm will be used as the basic component of the proposed scheme in section 4.2 to learn the weights of training instances in a WDNN classifier.

4.1 Learning the Best Operating Point in 2-Class Problems

A discrete classifier such as a classification tree only produces a class label for an input pattern. For a 2-class problem (with positive and negative class labels), given a test set of P positive and N negative labeled patterns, a classifier of this type generates a 2×2 confusion matrix (shown in Fig.2) representing the performance of the classifier. The accuracy of the classifier is defined as:

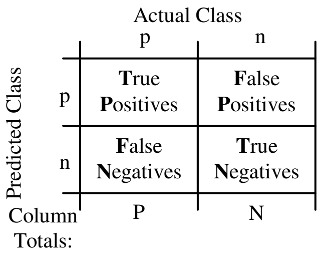
$$\text{Accuracy} = \frac{TP + TN}{P + N} = \frac{TP - FP}{P + N} + \frac{N}{P + N} \tag{8}$$

Many classifiers, such as Bayesian classifier or neural networks naturally assign a score $S(X_t)$ to each input pattern X_t (i.e., scoring classifiers). For example, naive Bayes classifiers output posterior probability distribution over classes. In this case, the score of a pattern for our 2-class problem can be defined as:

$$S(X_t) = \frac{pr(n,X_t)}{pr(p,X_t)} = \frac{pr(n,X_t)}{1-pr(n,X_t)} \tag{9}$$

Where $pr(p,X_t)$ and $pr(n,X_t)$ denote the estimated probabilities that the pattern X_t is of positive and negative class, respectively. With the above definition, the score is a numeric value (in the range 0 to ∞) expressing the degree that X_t is thought to be of negative class.

A scoring classifier can be converted to a discrete classifier by specifying a threshold on score. A pattern is classified as negative if its score is greater than the specified threshold and positive otherwise. In this way, the accuracy corresponding to each specified threshold can be calculated using (8).



		Actual Class p	Actual Class n
Predicted Class	p	True Positives	False Positives
Predicted Class	n	False Negatives	True Negatives
Column Totals:		P	N

Fig. 2. Confusion matrix for a discrete classifier

Having the relation between a threshold and corresponding accuracy of the classifier, the best threshold can be easily found by varying the threshold from 0 to ∞. Actually, it is sufficient to consider those thresholds such that classification of an instance changes from negative to positive. Based on this idea, an efficient algorithm for calculating the best threshold is given in [8, 17]. For this purpose, the patterns are ranked in ascending order of their scores (i.e., $S(X_1) < S(X_2) < \ldots . < S(X_{P+N})$). Considering any threshold between $S(X_K)$ and $S(X_{K+1})$, the first K patterns will be classified as positive and the remaining $P+N-K$ patterns as negative. In this way, a maximum of $P+N+1$ different thresholds should be examined to find the best threshold. The first threshold classifies everything as negative and the last threshold classifies everything as positive. The rest of the thresholds are chosen in the middle of two non-equal successive scores $S(X_K)$, $S(X_{K+1})$ in the list such that $S(X_K) \neq S(X_{K+1})$. The best threshold is simply the one that maximizes the accuracy (8) of the classifier. An algorithm to find the best threshold is given in Table 1. This algorithm receives a set of patterns and their scores as input and returns the best threshold (i.e. giving maximum classification rate) as output.

The important point is that the value of the best threshold (i.e., *best-th*) calculated using the algorithm of Table 1 can be used as the weight for positive class. That is, instead of classifying a pattern X_t as positive if $S(X_t) < best\text{-}th$, we can equivalently classify the pattern as positive if $best\text{-}th \times pr(p,X_t) > pr(n,X_t)$.

Table 1. Algorithm for finding the best threshold

```
Inputs: patterns $X_t$, scores $S(X_t)$
Output: the value of best threshold (best-th)
  current = number of misclassified patterns corresponding to the threshold of $th = 0$ (i.e.,
  classifying everything as $\overline{class\ T}$ )
  optimum = current
  best-th = 0
  rank the patterns in ascending order of their scores
  {assume that $X_k$ and $X_{k+1}$ are two successive patterns in the list}
  for each different threshold $th = (Score(X_k)+Score(X_{k+1}))/2$
    current = number of misclassified patterns corresponding to the specified threshold
    (i.e., all patterns $X_t$ having $Score(X_t) < th$ are classified as class T)
    if current < optimum then
      optimum = current
      best-th = th
    end if
  end for
  {assume that last is the score of last pattern in the list and $\tau$ is a small positive number}
  current = number of misclassified patterns corresponding to $th = last + \tau$ (i.e.,
  classifying everything as class T)
  if current < optimum then
    optimum = current
    best-th = th
  end if
return best-th
```

4.2 Learning Weights of Training Instances

For an M-class problem, assume that a training set Γ consisting of N labeled training patterns (i.e. $\Gamma=\{X_j, j=1, 2, ..., N\}$) is available. In this section, we propose an efficient algorithm that attempts to maximize the classification accuracy of the WDNN classifier on training data by learning the weights of instances in the training set.

In its basic form, the proposed algorithm is a hill-climbing search method. The algorithm starts with an initial solution to the problem (i.e., $\{w_k= 1, k =1, 2, \ldots, N\}$), and sequentially improves the solution by finding a neighbor solution that is better than the current one. A neighbor solution is different from the current solution in the value of just one parameter (i.e., the weight of one instance). Without the algorithm proposed in this section, many neighbor solutions should be examined (i.e., making the search process slow) to find a solution that is better than the current solution. This is due to the complexity of the problem and the fact that the optimization parameters (instance weights) are continuous.

The algorithm given below can provide neighbor solutions that are at least as good as the current solution. This algorithm, which is actually an extended version of the algorithm given in Table 1, finds the optimal weight of an instance assuming that the weights of all other instances are given and fixed. Note that, by optimizing the weight of one instance, the algorithm is indeed providing a neighbor solution that is better than (or at least as good as) the current solution.

We illustrate this algorithm to find the optimal weight of X_t ε Γ assuming that the weight of all other instances in Γ are given and fixed. Further, assume that X_k is a training instance of class *T*, where *T* ε {1, 2, …, *M*}. The optimal weight of X_k can be found using the following steps.

1. I = {}
2. Classify all training examples using $w_k = \infty$ (i.e., a very large positive number)
3. Classify all training examples using $w_k = 0.0$
4. Add to I those training examples that are classified correctly only in one of the previous steps (step 2 or 3).
5. Calculate the score of each training example $X_t \in$ I using the following measure.

$$S(X_t) = \frac{\max\limits_{1 \le j \le N} \left\{ w_j \cdot \mu(X_t, X_j) | X_j \in \Gamma \right\}}{\mu(X_t, X_k)} \tag{10}$$

6. The algorithm of Table 1 can now be used to find the best threshold (i.e. the best weight w_k of instance X_k). For this purpose, a 2-class situation is formed by considering class *T* as positive class and a class by merging all other classes as negative class.

The purpose of steps 1 to 4 is to identify those training examples (by collecting them in the set I) that their correct classification depends directly on the value of w_k.

Overall, the search for locally optimum solution starts with an initial solution to the problem and sequentially improving the solution using a hill-climbing approach. The above 6-step algorithm can be used to find all neighbor solutions (i.e., being different just in the value of one parameter) that are at least as good as the current solution. The usual hill-climbing search can be performed by sequentially replacing the current solution with the best neighbor solution. Finding all neighbor solutions can take a long time when the number of parameters (i.e., instance weights) is large. To speed up the search, in our implementation, the value of just one parameter is updated in each step of the hill-climbing search. That is, the search for locally optimum solution is conducted by optimizing each instance in turn assuming that the order of the instances to be optimized is fixed. The search terminates if no neighbor solution could be found that is better than the current one (or after a certain number of iterations). Obviously, with this modification, the algorithm is sensitive to the order of instances considered for optimization. In simulation results reported in section 5, we fix the ordering by sorting the instances based on their similarity to their nearest enemy instance.

$$\mu_{enemy}\left(X_k\right) = \max_{1 \le j \le N} \left\{ \mu\left(X_k, X_j\right) | \, class\left(X_j\right) \ne class(X_k) \right\} \tag{11}$$

Where, $\mu_{\text{enemy}}(X_k)$ is the similarity of instance X_k with nearest instance of enemy class, and *class*(X_k) is used to denote the true class of X_j. The instances in the training set are ranked in the descending order of measure (11) and optimized in that order.

When finding the best weight of an instance, it can happen that the set I is empty after step 4 of the above algorithm. This actually indicates that the classification rate on the training data cannot be improved by setting the weight of this instance. In other words, the instance can be pruned (i.e. by setting its weight to zero) without affecting the classification rate. Note that an instance can also be pruned if the value of *best-th* returned by the algorithm of Table 1 is zero. One key feature of the proposed scheme is pruning redundant instances visited during the learning process. The final set of instances usually contains much fewer instances contributed in the classification process. This feature is very useful since, the proposed algorithm can be considered as a serious instance reduction technique for the NN classifier.

5 Experimental Results

In our experiments, in order to compare our proposed method with the well-known classifiers in the BCI field, Support Vector Machine (SVM) [24] and Linear Discriminant Analysis (LDA) [25] are employed. The classification results are produced in the way by which one of the features (i.e. BP or FD) is applied to one of the classifiers (i.e. FLDA or SVM or WDNN), which referred to as the combinations of single feature classifier.

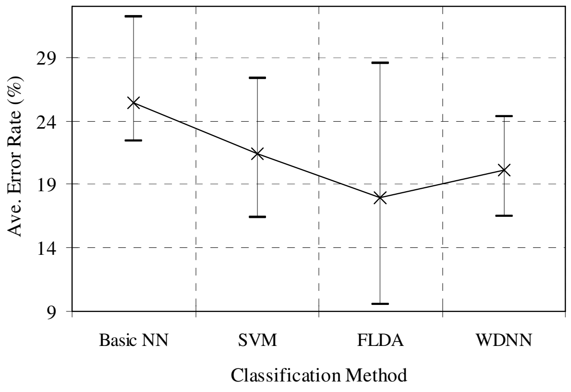
At first, EEG signals from four trained subjects (L1, O3, O8, and G8) were recorded. Out of 360 trials recorded for each subject, 240 trials were used in the train and the rest in the test phase. FD and BP features were extracted from the signals and basic NN, WDNN, SVM, and FLDA were used as the classifiers. In the train phase, significant features were selected using average accuracy on the validation set by ten times ten folds cross validation (10CV). In this way, a classifier is trained with the features of all trials in each 250 ms through the paradigm. Our paradigm is 8sec.; therefore, we have 32 different feature sets for all trials. To choose the best feature set, we calculate the error rate of the classifier by 10CV on training data. Classifiers, trained with the best feature set for each subject, were then used to classify test feature vectors.

Results of test data for th e subjects with all of the classifiers and features are shown in Table 2. It can be seen that the WDNN has a good compatibility with the FD feature in comparison with the BP. Results of combination of FD and WDNN, in three cases out of four, led to the better results in comparison with FLDA, SVM, and standard NN. Also, in the case O8, combination of WDNN and BP shows a supremacy compare to the other considered classifiers. Incidentally, in Table 3, number of stored instances in NN and WDNN are compared which shows a drastic instance reduction performed while generalization property is increased dramatically. The reason is that in the NN method all trained samples (240) are considered for measuring the class label of an unknown pattern but in the WDNN method, a lot of trained samples have zero weights, therefore, they are removed automatically from the training set.

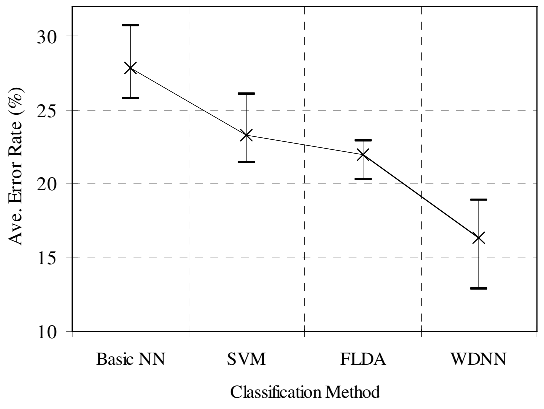
In Fig. 3, the average error rates of each method on every four subjects are shown. For each classification method, minimum and maximum error rates on the whole cases under investigation is also depicted. In Fig. 3(a), the obtained results are shown in case of using BP features for classification and Fig. 3(b) displays the same statistics in case of using FD features.

Table 2. Test error rates for different subjects using different features and classifiers

Subject	Feature	Basic NN	SVM	FLDA	WDNN
L_1	FD	32.16	27.38	28.57	24.34
	BP	28.65	23.81	20.24	18.67
O_3	FD	23.98	19.8	9.59	17.39
	BP	30.65	26.03	21.92	12.83
O_8	FD	22.43	21.90	17.14	16.48
	BP	25.72	21.90	22.86	12.82
G_8	FD	23.26	16.43	16.43	22.37
	BP	26.29	21.43	22.86	18.86



(a)



(b)

Fig. 3. Average error rates on four subjects obtained by applying compared methods and the maximum and minimum error rates achieved by them are shown in (a) for BP features and in (b) for FD

Table 3. Number of training instances stored in the test phase for NN and WDNN methods

Subjects	Feature	Basic NN	WDNN
O_3	FD	240	29
	BP	240	34
O_8	FD	240	30
	BP	240	20
L_1	FD	240	28
	BP	240	34
G_8	FD	240	17
	BP	240	19

As it can be seen in Fig. 3(a), in case of using BP features, WDNN performs better than Basic NN and SVM, but FLDA is the best method in this case. Note that WDNN is still more reliable, since the deviation of error rate is less than FLDA method. In Fig. 3(b), in case of using FD features, it can be seen that the proposed WDNN method has the highest classification accuracies among the compared classifiers. Then, it can predict between the left and right imagery tasks with less error rates. Note that in this case, maximum error rate resulted by applying WDNN is still less than minimum error rates occurred by the other methods presented in this article.

6 Conclusion

In this paper, an innovative approach has been proposed in order to improve the classification rate on the left and right imagery tasks in the cue-based BCI. NN is a traditional method but it is still a powerful method in various applications. In this research, a new version of NN is developed. The noisy instances degrade the performance of the nearest neighbor. NN also considers the importance of all the stored instances the same, but in fact, their relevancy is not the same. This paper presents a novel learning algorithm which is used to assign a local weight to each stored instance, which is then contributed in distance measure, with the goal of improvement in generalization ability of the basic NN. The learning algorithm optimizes the instance weights based on the classification accuracy. The presented scheme achieves two purposes at the same time. The classification rate is improved by adjusting a weight for each instance and considering it while calculating distance measure. It is also faster in predicting between the left and the right tasks for a new subject introduced to it. Since, majority of training instances are removed by assigning a zero weight to them and will not contributed in classification task. In order to evaluate our method, it has been applied on the BP and FD features of two imagery tasks of four subjects (L1, O3, O8, and G8) participated in this study. The results showed that the proposed method is effective to achieve higher accuracy to choose between the left and right tasks.

References

1. Pfurtscheller, G., Neuper, C.: Motor imagery and direct brain-computer communication. In: Proc. IEEE, pp. 1123–1134. IEEE Computer Society Press, Los Alamitos (2001)
2. Pfurtscheller, G., Lopes, S.: Event related desynchronization: Hand book of electroenceph. And clininical Neurophisiology. Revised edition, vol. 6. Elsevier, Amsterdam (1999)
3. Bozorgzadeh, Z., Birch, G.E., Mason, S.G.: The LF-ASD brain computer interface: on-line identification of imagined finger flexions in the spontaneous EEG of able-bodied subjects. In: IEEE Intern. Conf. Acous. Speech proc., vol. 6, pp. 2385–2388 (2000)
4. Boostani, R., Moradi, M.H.: A new approach in the BCI research based on fractal dimension as feature and Adaboost as classifier. Journ. Neural Eng. 1, 4 (2004)
5. Boostani, R., Graimann, B., Moradi, M.H., Pfurtscheller, G.: A Comparison Approach toward Finding the Best Feature and Classifier in Cue-Based BCI. Journ. Medic. Bio. Eng. Comp. 6 (2007)
6. Schlögl, A., Flotzinger, D., Pfurtscheller, G.: Adaptive Autoregressive Modeling used for Single-trial EEG Classification. Biomediz. Tech. 42, 162–167 (1997)
7. Graimann, B., Huggins, J.E., Levine, S.P., Pfurtscheller, G.: Toward a direct brain interface based on human subdural recordings and wavelet-packet analysis. IEEE Trans. Biomed. Eng. 51, 954–962 (2004)
8. Pfurtscheller, G., Neuper, C., Schlogl, A., Lugger, K.: Separability of EEG signals recorded during right and left motor imagery using adaptive autoregressive parameters. IEEE Trans. Rehab. Eng. 6, 316–328 (1998)
9. Haselsteiner, E., Pfurtscheller, G.: Using time-dependent neural networks for EEG classification. IEEE Trans. on Rehab. Eng. 8, 457–463 (2000)
10. Kalcher, J., Flotzinger, D., Pfurtscheller, G.: A New Approach to a Brain-Computer-Interface (BCI) based on Learning Vector Quantization (LVQ3). In: Proceed. Ann. Intern. Conf. IEEE, vol. 4, pp. 1658–1659 (1992)
11. Flotzinger, D., Pregenzer, M., Pfurtscheller, G.: Feature selection with distinction sensitive learning vector quantisation and genetic algorithms. IEEE Intern. Conf. Comp. Intell. 6, 3448–3451 (1994)
12. Deriche, M., Al-Ani, A.: A new algorithm for EEG feature selection using mutual information. In: IEEE Intern. Conf. Acous. Speech, Signal Proc. ICASSP, vol. 2, pp. 1057–1060 (2001)
13. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification. IEEE Trans. on Info. Theo. 13, 21–27 (1967)
14. Dasarathy, B.V.: Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. IEEE Computer Society Press, Los Ala. CA (1991)
15. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood component analysis. Neur. Info. Proc. Sys (NIPS) 17, 513–520 (2004)
16. Weinberger, K., Blitzer, J., Saul, L.: Distance metric learning for large margin nearest neighbor classification. In: Weiss, B.S., Platt, J. (eds.) Advances in Neural Information Processing Systems, p. 18 (2005)
17. Wang, J., Neskovic, P., Cooper, L.N.: Neighbourhood selection in the k-nearest neighbor rule using statistical confidence. Patt. Recog. 39, 417–423 (2006)
18. Wang, J., Neskovic, P., Cooper, L.N.: Improving nearest neighbor rule with a simple adaptive distance measure. Pattern Recognition Letters 28, 207–213 (2007)
19. Xiao-Yuan, J., David, Z., Yuan-Yan, T.: An improved LDA Approach. IEEE Trans. Sys. Man Cyber. 34, 5 (2004)

20. Esteller, R.: Detection of Seizure Onset in Epileptic Patients from Intracranial EEG Signals. Ph. D. thesis, School of Electrical and Computer Engineering Georgia Institute of Technology (2000)
21. Higuchi, T.: Approach to an Irregular Time Series on the Basis of Fractal Theory. Physica D 31, 277–283 (1988)
22. Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Researchers, Technical Report HPL-2003-4, HP Labs (2003)
23. Lachiche, N., Flach, P.: Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In: 20th Intern. Conf. Machine Learning (ICML'03), pp. 416–423 (2003)
24. Vapnic, V.N.: Statistical learning theory. John Wiley and Sons, New York (1998)
25. Fukunaga, K.: Introduction to Statistical Pattern Classification. Academic Press, San Diego, Calif. (1999)