# Non-identity Learning Vector Quantization applied to evoked potential detection

Nanying Liang, Laurent Bougrain

# NON-IDENTITY LEARNING VECTOR QUANTIZATION APPLIED TO EVOKED POTENTIAL DETECTION

Nanying Liang
INRIA / LORIA
Nancy, France
nanying.liang@loria.fr

Laurent Bougrain
INRIA / LORIA
Nancy, France
bougrain@loria.fr

**ABSTRACT**
This article presents a new computational intelligence technique for pattern recognition of graphic elements (e.g. event-related potential, auditory evoked potential, k-complex, spindle) embedded in electro-encephalographic signals. More precisely, we have extended the learning vector quantization (LVQ) algorithm by Kohonen to non-identity assignment to robustly detect evoked potentials in a noisy electro-encephalographic signals for brain-computer interfaces. The improved LVQ is obtained by optimizing its assignment layer through the minimum norm least square algorithm, the same scheme found in Extreme Learning Machine (ELM). The proposed LVQ is evaluated using the Wadsworth BCI datasets on P300 speller. The experimental results show that the proposed LVQ improved the performance with less computational units.

**KEY WORDS**
Learning Vector Quantization, Extreme Learning Machine, nonlinear separable problem, brain-computer interface, evoked potential detection.

## 1 Introduction

Oddball paradigms are used by brain-computer interfaces to generate event-related potentials (ERPs) on visual or auditory targets on which the user focusses his attention. Each target is associated to a specific action. The well-known P300 speller is based on this principle [1]. The main problem is to be able to detect ERP in a noisy electroencephalographic signal recorded on human scalp. Averaging several responses is usually necessary to detect ERPs. Thus, methods based on "EP templates" are potentially interesting to detect ERP. Templates can be obtained using averaging techniques and clustering methods such as learning vector quantization.

Learning vector quantization (LVQ) algorithms perform a supervised learning for a classification task. The basic idea and some variants have been proposed by Kohonen [2]. The principle is to cluster input data using a competitive learning without any spatial relationship between codebook vectors. Each cluster is pre-assigned to a specific class. Thus, when a new pattern should be classified, the method determines which cluster the pattern belongs to and the assigned class. The main interests for this machine learning technique are both the robustness and the interpretability. However for large-scale complex problems, its performance can be further improved by optimizing the way to combine the templates in its assignment stage rather than just selecting one of them as the decision. More precisely, it could be interesting to overcome the simple identity function to determine classes from clusters. Thus, a non-identity learning vector quantization will be described in this article to obtain a more powerful linear assignment. This improvement will be done using an extreme learning machine algorithm.

Extreme learning machine (ELM) algorithm has a fast training speed by just tuning its layer weights and left fixed its randomly chosen input weights during the training procedure. This normally requires to use more layer units than neural network algorithms which also tune the input weights.

Thus, in this article we will combine both LVQ and ELM algorithms to propose a new training algorithm named, LVQ-ELM, in which LVQ focuses on tuning the input weights, and ELM on tuning the layer weights. The proposed algorithm should be able to improve the performance of LVQ, and at the same time, uses less neurons than ELM.

The following sections will describe both the LVQ and ELM algorithms (section 2.1 and 2.2 respectively) and how to combine them into a new algorithm named LVQ-ELM (section 2.3). Then, we will present and discuss experimental results obtained using the Wadsworth BCI datasets on P300 speller (section 3). Finally, we will resume and conclude on this work (section 4).

## 2 Methods

### 2.1 Learning Vector Quantization

**LVQ1** Learning vector quantization (LVQ) algorithms proposed by Kohonen [2] are supervised clustering methods designed for classification. A set of codebook vectors $V^i$ with the same dimension than input $X$ store the cluster center-based vectors. Each cluster is pre-assigned to a class. Several clusters can be assigned to the same class. The percentage of clusters assigned to one class is usually proportional to the percentage of training samples of this class.

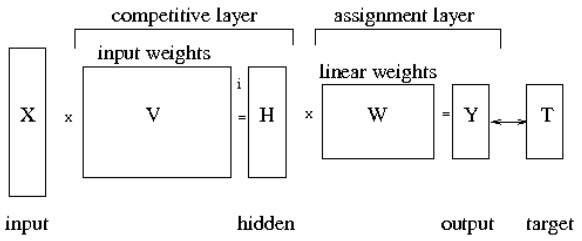


Figure 1. Generic architecture for LVQ algorithms

The architecture of these algorithms (Fig. 1) can be defined as a two-layer neural network: a competitive layer and an assignment layer. Each neuron of the competitive layer (or hidden layer) corresponds to a cluster and is characterized by a codebook vector $V^i$. A classic competitive step is used to determine which neuron is the closest one to the current input using the following formula: $c = arg\max_i\{||X - V^i||\}$. The closest neuron is regarded as the winner neuron and others as loser neurons. The output vector H of the competitive layer are computed as follow: $h_i = winner - take - all(||X - V^i||)$, where the $winner - take - all(.)$ function produces 1 as the output of the winner neuron, and 0 for other neurons. These competitive layer's classes are transformed into target classifications through the layer weights $W$.

Each neuron of the second layer corresponds to a specific class. Thus, the weights of the second layer relate clusters with classes. In the standard first version LVQ1, each cluster is exclusively assigned to one class. For example, suppose neurons $i-1, i, i+1$ in the competitive layer are assigned to class $k$, then these competitive neurons will have W weights of 1 that link to the class $k$ output neuron and 0 to other class output neurons. The output $Y$ of LVQ1 can be written as:

$Y = W * H = W * winner - take - all_i(||X - V^i||)$

Thus $Y_k$ produces 1 if any of neurons $i-1, i, i+1$ wins the competition. $Y$ is compared to the target vector $T$ to compute the accuracy.

In the learning procedure of LVQ1, W is fixed and only the codebook vector of the winner neuron $c$ is updated according to the following rules:

$V^c(t+1) = V^c(t) + \alpha(t)[X(t) - V^c(t)]$ if $X$ and $V^c$ belong to the same class

$V^c(t+1) = V^c(t) - \alpha(t)[X(t) - V^c(t)]$ if $X$ and $V^c$ belong to different classes

where $\alpha$ is the learning rate. The application of LVQ1 requires to predefine the value of learning rate, the number of competitive neurons and the number of epoches.

## 2.2 Extreme Learning Machine

Extreme learning machine (ELM) proposed by Huang et al. [3] is for training single hidden layer feedforward neural networks (SLFNs). [4] proved that SLFNs with randomly generated addictive or radial basis function (RBF) neurons can universally approximate any continuous functions in a compact subset of the Euclidean space. Based on this theory, only the output layer weights (W) is analytically computed in the training procedure. In the ELM algorithm, W is solved by the minimum norm least square solution:

$W = H^{\dagger}T$

where $H$ is the hidden layer output vector, and $T$ is the target vector. The symbol † denotes the pseudo-inverse operation. The networks with the smallest norm of output weights likely achieve the best generalization performance [5]. Simulations of ELM had been carried out with real-world applications [4]. The simulation results showed that ELM, with extremely fast training speed, obtained equal or better generalization performance than support vector machine (SVM) [6], which is well-known for its generation capability in the machine learning area. However, the size of SLFNs (i.e., the number of hidden neurons) trained by ELM is always larger than the algorithms that both optimize the input and output weights simultaneously, and thus requires more computing sources especially in the large-scale complicate applications.

## 2.3 LVQ-ELM

From the previous two sections, we observe that, in the training procedure, LVQ only updates the codebook vectors $V^i$ and left the assigments $W$ fixed; While ELM optimizes $W$ and let $V^i$ be randomly chosen and fixed. LVQ works well in linear applications as ELM. However in non-linear applications, the generalization performance of LVQ is lower than ELM; While ELM requires large number of hidden neurons and leads to out of memory problems. To compensate these two algorithms to preserve the generalization capability while without increasing the computing resources, we propose a new training algorithm named LVQ-ELM. The principle of LVQ-ELM is to compute the $V^i$ using the scheme of LVQ1 and to optimize $W$ through the same minimum-norm least-squares solution as ELM. To derive the algorithm of LVQ-ELM, the output of LVQ networks is repeated here:

$Y = W * winner - take - all_i(||X - V^i||)$

If one directly applies the minimum-norm least-square solution of W to the above equation, it can not improve the performance of LVQ due to the competitive function filtering out the information necessary for fine tuning the values of W. We propose to place the competitive operation after the weighted negative distance as follows:

$Y = winner - take - all_i(W||X - V^i||)$

The output does not change. Now, we can introduce the minimum-norm least-square solution to compute W as:

$$W = \begin{bmatrix} ||X - V^1|| \\ \vdots \\ ||X - V^i|| \end{bmatrix}^{\dagger} T$$

where $T$ is the target output vector and $V^i$ are the codebook vectors optimized using the LVQ1 rule.

Depending on the position where the minimum-norm least-square solution is introduced into LVQ1 rule, we propose three possible implementations of LVQ-ELM. The semi-code of one possible implementation is given as follows:

initialize $V^i$, for each $i$, as the midpoints of the input X, and W as binary values with percentage of 1s according to the percentage of the training samples of each class:

**for** each epoch **do**
  randomly shuttle the order of the training patterns
  **for** each pattern (X,T) **do**
    $\forall_i, h_i = ||X - V^i||$
    $Y = winner - take - all(H^T W)$
    $c = argmin\_i(h_i)$
    $predicted\_class = argmax\_k(y_k)$
    $target\_class = argmax\_k(t_k)$
    **if** $predicted\_class = target\_class$ **then**
      $V^c = V^c + \alpha(X - V^c)$
    **else**
      $V^c = V^c - \alpha(X - V^c)$
    **end if**
  **end for**
  $W = (||X - V^i||)^\dagger T$
  check the stopping condition
**end for**

The above implementation updates $W$ at each epoch. Another quite similar implementation of LVQ-ELM is to update $W$ after the LVQ1 completes its optimization on $V^i$. However, we suggest to update $W$ at each epoch because it is necessary to examine the stopping condition to know where the algorithm stops at the end of each epoch. The third possible implementation is to update $W$ after the presentation of each pattern. We further refer to this implementation as LVQ-OSELM (for OSELM, refer to [7]). OSELM itself is suited for online sequential learning situation, that means, at each time, one pattern is presented and learned by algorithm, and will be discarded from the computer memory. From this point of view, the advantage of OSELM is useless for the practical implementation of LVQ-OSELM, and therefore, we will not adopt it in our simulation either.

# 3 Experimental Results

## 3.1 Wadsworth BCI dataset

The P300 speller data set from Wadsworth BCI [8] is used here for the purpose of algorithm evaluation. The data set contains two subjects, A and B. For each subject, there are 85 letters for training and 100 letters for testing. For each letter, the recording consists of 15 epoches, and within each epoch, there are 12 flashings. At each time, flashing is randomly chosen to highlight one row/column. The virtual keyboard contains six rows and six columns consisting of a 6x6 grid representing 26 characters, 9 numbers and one dash character. We are interested in measurements of a one-second window starting from the onset of the flashing, which consists of 240 samples at a sampling rate of 240Hz. In case the highlighted row/column contains the target letter, we refer to it as the EP response; otherwise the background EEG activities. We first use the raw EEG measurements in time domain and normalize the amplitude into range between 0 and 1 and focus on one channel Cz which can observe P300 component well. The task of the algorithms is to discriminate two EP responses from the other background EEG activities for each epoch, and then to predict the spelled letter. We then repeat the experiment but using all 64 channels in the dataset. Due to limited RAM memory available, we need further preprocess the raw EEG measurements by using a moving averaging filter (moving window size equals to 13) followed by a subsampling operation (subsampling factor equals to 13).

## 3.2 Model Selection

Before performance evaluation, one need to determinate the model parameters, for example, the number of hidden neurons in ELM, and for LVQ1 and LVQ-ELM, the number of hidden neurons, the number of epoches and the learning rate. The procedure for selecting these parameters is called model selection and is implemented using the train and validate method here. Hence, we need to further divide the training data set of 85 letters into two subsets, the first 75 letters for training and the left 10 letters for validation. The search procedure for ELM is in a one-dimension space and a three-dimension space for LVQ1 and LVQ-ELM. In order to facilitate the search procedure for LVQ1 and LVQ-ELM, the number of epoches is decided inside the learning procedure by examining the changes of validate accuracy to tell at which point the training procedure should be stopped. Further, we decompose the searching space into two one-dimension spaces. First lets fix the learning rate at a reasonable small value, and only optimize the number of hidden neurons. After finding out the optimized number of hidden neurons, we vary the values of learning rate in a predefined possible range.

## 3.3 Performance Evaluation

Table 1 summarizes the results of model selection and the correctly classified accuracy based on EEG measurements from channel Cz. The values of model parameters are optimize through the procedure as described in foregoing subsection. Here LVQ1 needs less hidden neurons than ELM. Once the model parameters are optimized, the algorithms are evaluated using the testing data set to estimate their generalization performance. The generalization performance in our simulation is referred as the percentage of correctly classified letters out of 100 testing letters and averaged over 30 trials of simulations. LVQ-ELM achieves the best performance in both subjects but the improvement is not ob-

| Dataset | Method ELM | LVQ1 | LVQ-ELM |
|---|---|---|---|
| Subject A | 40.60% | 40.43% | 42.30% |
| | 240-80-2 | 240-20-2 | 240-80-2 |
| | | lr=0.1 | lr=0.005 |
| Subject B | 25.96% | 25.86% | 26.63% |
| | 240-40-2 | 240-12-2 | 240-12-2 |
| | | lr=0.1 | lr=0.1 |

Table 1. In the first row, the percentage of correctly classified letters for the test set are averaged over 30 trials and on the base of EEG measurements from channel Cz and below, the optimized model parameters are represented as the architecture of the networks and the value of learning rate.

| Dataset | Method ELM | LVQ1 | LVQ-ELM |
|---|---|---|---|
| Subject A | 48.90% | 32.10% | 87.00% |
| | 1152-700-2 | 64X[18-8-2] | 64X[18-8]-2 |
| | | lr=0.1 | lr=0.1 |
| Subject B | 50.80% | 25.83% | 96.00% |
| | 1152-700-2 | 64X[18-8-2] | 64X[18-8]-2 |
| | | lr=0.1 | lr=0.1 |

Table 2. In the first row, the percentage of correctly classified letters for the test set are averaged over 30 trials and on the base of EEG measurements from 64 channels and below, the optimized model parameters are represented as the architecture of the networks and the value of learning rate.

vious for this simulation just based on EEG measurements from one channel.

Therefore we repeat the above experiment but based on the measurements from all channels. The simulation results are then summarized in Table 2. For this case, the number of input features increases. Due to limited RAM memory available, we have to filter the signal and then take a subsample in order to keep as much information about the EEG measurements as possible while reducing the number of input features from 64*240 to 64*18. To extend to multi-channel case, for ELM, the input vectors from 64 channels are concatenated as one input, therefore its input dimension is of 1152. For LVQ1, we let each LVQ model represent one channel. LVQ-ELM is the combination of LVQ1 and ELM: we let each independent competitive layer to learn the morphology information from each channel measurements same as LVQ1; while its assignment layer is similar with ELM to have each output neuron fully linked to the competitive neurons of all independent layers. For this case, it is clear to see LVQ-ELM obtains the best testing accuracy using less hidden neurons than ELM.

## 4 Conclusion

The experimental results show that LVQ-ELM improves the performance of LVQ by optimizing its layer weights using minimum-norm least squares method using the same scheme found in ELM. It is also found that LVQ-ELM may reduce the number of neurons that ELM required to achieve the same or better performance. Such advantage is even more evident in large-scale application. The performances obtain by our method are similar to the best algorithms used on Wadsworth BCI dataset[1]. So, from a practical BCI view of point, it will be interesting to investigate how this approach supports a reduction of the number of epochs in order to increase the bit rate and make the patient more comfortable with this equipment.

Moreover, cluster center-based methods are interesting for knowledge extraction. So, the method of LVQ-ELM will also be benefited for other problems such as, in the medical domain, auditory evoked potential detection for automated newborn hearing screening and k-complex detection for automatic sleep scoring.

## References


[1] L.A. Farwell and E. Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510–523, 1988.

[2] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 78, pages 1464–1480, 1990.

[3] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew. Extreme learning machine: A new learning scheme of feedforward neural networks. In *Proceedings of International Joint Conference on Neural Networks*, volume 2, pages 985–990, Budapest, Hungary, 2004.

[4] G.-B. Huang, L. Chen, and C.-K. Siew. Universal approximation using incremental constructive feedforward networks with random hidden neurons. *IEEE Transactions on Neural Networks*, 17(4):879–892, 2006.

[5] P. L. Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, 44(2):525–536, 1998.

[6] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[7] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan. A fast and accurate on-line sequential learning algorithm for feedforward networks. *IEEE*


[1] http://ida.first.fraunhofer.de/projects/bci/competition_iii/results/index.html

*Transactions on Neural Networks*, 17(6):1411–1423, 2006.

[8] D. J. Krusienski and G. Schalk. *Wadsworth BCI Dataset (P300 Evoked Potentials)*. BCI Competition III Challenge, 2004.