ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 203-208

A New Dynamic LVQ-based Classifier and Its Application to Handwritten Character Recognition*

Sergio Bermejo†, Joan Cabestany and Magda Payeras

AHA Group
Department of Electronic Engineering, Universitat Politècnica de Catalunya (UPC).
Gran Capità s/n, C4 building, 08034 Barcelona, Spain
e-mail: sbermejo@eel.upc.es

Abstract- In this paper we present a new dynamic learning strategy in LVQ's framework that consists of a growing process and a posteriori pruning process. This schema allows building a family of dynamic LVQ learning systems. To evaluate this proposal, we have employed LVQ3 as a core of the learning system. We have done an empirical analysis of this system (DLVQ3) in order to characterize its parameters and to compare it with non-dynamical LVQ algorithms. Finally we present results on applying PCA_{64}+DLVQ3 to upper and lower handwritten character recognition, obtaining on average test classification error (with no rejection) 7.22% and 15.275 % respectively in front of 9.16% and 17.01% achieved with PCA_{64}+ non-dynamic LVQx's of similar size, 14.7% and 23.1 % with PCA_{64}+PNN [3] & 10.1% and 20.3% with PCA_{128}+MLP [3].

1.Introduction

LVQ algorithms are a kind of vector quantizers that are strictly meant for a statistical pattern classification, because of "its only purpose is to define *class regions* in the input data space" (p.203; [1]). Starting on a training pattern set and by means of simple supervised learning mechanisms repetitively applied over this set they finally obtain a number of prototype (or codebook) vectors that allow to build class borders according to the nearest neighbor rule. In this way, they define class regions over the input space.
Generalization in these systems, which we measure with patterns (test data) never used in training, depends on:

- Factors associated to codebook, as the number of codebook vectors assigned a priori to each class and its initial values
- Factors associated to learning algorithm, as the used adaptive mechanisms, the learning rate fixed by the algorithm parameters and the chosen stopping criteria.

Thus, correct application of LVQ algorithms requires always bearing in mind these factors, and because of these, like in other learning systems, it turns into an art. A solution to this problem has been given by LVQ_PAK [2]. In this software package, we

* This work has been funded by spanish CICYT action TIC96-0889
† granted by CIRIT action FI97/0621

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 203-208

can find, among others, programs to initialize codebooks (*eveninit* y *balance*) and a learning strategy based on use of a cascade of LVQ algorithms ordered from lesser to greater complexity. With this strategy, they pursue to refine gradually the initial codebook to a solution with a better generalization. When we apply it over training sets extracted from real problems, we can observe that sometimes, although we have initialized suitably a codebook with an optimal size, this learning system cannot learn to define correctly all class regions. In other words, it can learn to form better some class borders than others and thus it generalizes better in some classes than others. Without to make a detailed analysis of the reasons for this behavior, since codebook vectors define class regions, we can agree that:

- Either number of vectors to define each class is not the same
- Or vectors has not been properly placed due to learning dynamics

In this paper, we propose a dynamic LVQ learning algorithm that tries to solve this problem. Since it allows incrementally placing new codebook vectors depending on the codebook failure to classify each class well, our system tries to define all class regions with identical accuracy. So this dynamic learning schema gets a system which intends to improve global generalization via training error equalization between classes.
The rest of this paper is organized as follows. In section 2, we introduce our dynamic learning schema. Section 3 presents the preliminary results of several experiments made in the recognition of upper handwritten characters, in order to validate our algorithmic proposal. In section 4, we apply dynamic LVQ3 (our dynamic learning schema plus LVQ3) in the recognition of upper and lower handwritten characters and also compares our results, first with non-dynamical LVQ algorithms and then, with other neural approaches found in [3]. Finally, in section 5 preliminary conclusions are given.

2. Proposed Dynamic LVQ Learning Algorithm.

The basic philosophy of our dynamical LVQ system is based on early proposals [4]. The core of our system there is a (non-dynamical) LVQ learning algorithm that receives the initial codebook. This algorithm refines the codebook and ends after a predefined number of iterations. Then, the system decides with a well-defined criterion to finish training with a pruning process or to continue. In the latter case, our system inserts on the codebook a certain number of new vectors. In typical constructive LVQ systems, new codebook vectors are placed in class regions where training patterns are bad classified. As example, we can see Dynamic LVQ [5]. This algorithm inserts for every class a new codebook vector that is the mean of bad-classified class training patterns. Our proposal is a bit different, inspired by Fritzke's work in codebook construction of LVQ quantizers [6] and follows OSLVQ proposal [7]. Since codebook vectors define class regions, *why don't we insert new vectors close to those which its failure is the worst?* With this strategy when training stops, we could get a system that tries to define all classes with the same accuracy.
The proposed dynamic learning algorithm follows: first of all, a codebook arrives at our system. This is passed through a non-constructive LVQ algorithm during a number of epochs fixed by the parameter EPUP (*ep*ochs to *up*date). If meanwhile LVQ algorithm

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 203-208

is executed, the number of global iterations (length) is greater than RLEN (*r*unning *leng*th), algorithm stops returning current codebook (STOPPING CONDITION #1) passed through a pruning process. Once EPUP epochs have finished, the LVQ algorithm presents a new codebook due to training. If new codebook does not improve classification (STOPPING CONDITION #2) then system stops returning the last codebook pruned. Otherwise it inserts, regulated by BETA, no more than MAV (*m*aximum *a*dded *v*ectors) new codebook vectors in regions where current codebook classifies worst. Now, we are going to define this stopping criterion, the heuristic to place new codebook vectors and the pruning process.

STOPPING CONDITION 2 is defined in the following way:

$$SC\ \#2[n+1] = \{E_T(C_n) < E_T(C_{n+1})\} \& \left\{ length\,[n+1] > \frac{2}{3} RLEN \right\}$$

where $C_n=\{w_j\}$ is the codebook at time n and $E_T(C_n)$ is the number of misclassified patterns by a 1-NN classified based in Euclidean distance that uses C_n as its prototype vector set. This stopping criterion pretends algorithm to stop when new vectors placement doesn't improve classification error over training set. The right side of condition was inserted to avoid stopping on a local minimum of classification error function. We suppose that only we can find a global minimum in final training stage. It is clear that this hypothesis can only be true in certain restricted conditions so stopping criteria #2 must be revised in future research.

The generation of new codebook vectors is computed in the following way:

1. Compute $Cerr_{n+1}$, being this codebook of maximum size MAV, a set of those codebook vectors that belongs to C_{n+1}:

$$\{\vec{w}_j\} = C_{n+1} \supset Cerr_{n+1} = \{\vec{w}_i\};\ |Cerr_{n+1}| \le MAV$$
$$E(\vec{w}_i) \ge E(\vec{w}_j) \quad \forall j = 1 \ldots |C_{n+1}|,\ \forall i = 1 \ldots |Cerr_{n+1}|$$

where $E(w_i)$ is the number of classification errors due to codebook vector w_i. In other words, $E(w_i)$ is the number of input patterns which class is different from vector w_i that falls into its region of influence (ROI). $E(w_i)$ can be expressed as

$$E(\vec{w}_i) = \sum_{l=1}^{c} E_{cl}(\vec{w}_l)$$

where c is the number of classes and $E_{cl}(w_i)$ is the number of class cl classification errors due to codebook vector w_i or (in other words) the number of class cl vectors that fall in w_i 's ROI, with w_i belonging to another class.

2. For each w_i that belongs to $Cerr_{n+1}$:

2.1. Compute the input pattern set $T''_{n+1}(\vec{w}_i)$. This set is defined in the following way:

$$T''_{n+1}(\vec{w}_i) = \{\vec{x}_m\} = \arg\underset{T}{\{max\ |T^l_{n+1}(\vec{w}_i)|\}};\quad |T^l_{n+1}(\vec{w}_i)| = E_{cl}(\vec{w}_i)$$

where $T^l_{n+1}(\vec{w}_i)$ represents those patterns of training set that belong to class cl and fall into w_i 's ROI with w_i belonging to another class. It is clear that this set size is equal to the number of class cl classification errors due to codebook vector w_i.

2.2 Compute mean vector x_{prod} of set $\{x_m\}$as:

$$\vec{x}_{prod} = \frac{1}{E_{cl}(\vec{w}_i)} \sum_{m=1}^{E_{cl}(\vec{w}_i)} \vec{x}_m$$

2.3.Create a new codebook vector with these values:

$$\vec{w}'_i = \vec{w}_i + BETA\ (\vec{x}_{prod} - \vec{w}_i)$$
$$class\ [\vec{w}'_i] = class\ [\vec{x}_m]$$

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 203-208

Once the growing process is finished, the pruning process removes those codebook vectors that are not used for *correct* classification of the training set samples. With this simple process added at the end of training, we hope to eliminate either redundant codebook vectors that contribute to form class regions rightly, or those useless (if there are any) that only contribute to misclassify.

3. Empirical Analysis of DLVQ3

In above section, we have presented a dynamic learning strategy that can use any kind of LVQ algorithm as a core. To evaluate this proposal, we embedded LVQ_PAK's LVQ3 in our learning system and called the resultant global system Dynamic LVQ3 (DLVQ3). In this section, we present empirical evaluation of DLVQ3 tested on upper handwritten data set.

3.1. Handwritten Database and its preprocessing.

This handwriting data set (and the lower set used in section 4) can be found in [3] in directories: train/hsf_4, train/hsf_6 and train/hsf_7. We have chosen the last one as test set (11992 uppercase binary images) and the two others as training set (24420 images). These images (32x32 pixels) have been preprocessed with a Principal Component Analyzer that extracts 64 components from each image (NIST's mis2evt utility). The correlation matrix of PCA was computed from training set.

3.2. Experiments.

We have done three experiments in order to characterize Dynamic LVQx parameters and to compare it with non-dynamical LVQ systems.

3.2.1. Codebook Initialization and default parameters in DLVQ3.

Every experiment related to DLVQ3 applies in cascade first *eveninit, balance, olvq1, lvq1 & lvq2* programs from LVQ_PAK to obtain a DLVQ3's initial codebook. The parameter values used in the calling process of these programs follow. Eveninit: noc=<initial codebook size>; olvq1: rlen=90000; lvq1: rlen=90000, alpha=0.01; lvq2: rlen=90000, alpha=0.01, win=0.3. Default parameters values used throughout experiments, those related with LVQ3 algorithm embedded in DLVQ3, are the following: alpha=0.01, win=0.3, epsilon=0.1.

3.2.2.Experiment 1.

DLVQ3 was trained with two MAV values (2 and 10) for several initialcodebook sizes (50, 150, 234 and 280), BETA values (0.2, 0.5, 0.7, 0.9 and 1.0) and EPUP values (1, 5 and 20). We applied in cascade 4 times DLVQ3 with RLEN=500000. A total of 480 DLVQ3 simulations were performed.

3.2.3.Experiment 2.

DLVQ3 was trained with a low initial codebook size (50) for two values of MAV (2 and 5) and BETA (0.2, 0.5, 0.7, 0.9 and 1.0). We applied in cascade 4 times DLVQ3 with RLEN and EPUP having these values: (100000, 1), (500000,5), (500000,10) and (15000000, 20).

3.2.4.Experiment 3.

We compare best results obtained in experiment 1 with results from applying in cascade LVQ_PAK learning systems (*eveninit->balance->olvq1->lvq1->lvq2->lvq3*) to obtain a codebook of similar sizes. Parameter values are the same of those appeared in section 3.2.1 except RLEN (2500000 in lvq3 and 150000 otherwise).

3.3. Preliminary Results.

Because of brevity of this paper, we only report a list of conclusions extracted from simulations:

ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 203-208

- Learning rate increases smoothly as BETA does and is more stable in time (e.g. no local minim of training error) with low values of EPUP.
- Exists an optimal value of BETA (<=1.0) in terms of generalization (test) error and this value is affected mainly by EPUP.
- The number of vectors pruned mainly depends on MAV, although EPUP plays a secondary role (High values of EPUP=>less redundancy). If MAV increases redundant vectors does, even when EPUP is high.
- If EPUP and RLEN vary dynamically on training, we can obtain smaller codebooks.
- Generalization is better in DLVQ3 than in LVQ's system with the same codebook size, even when DLVQ3 begins on a small codebook.
- Variance of class training error (training error for each class) is higher in LVQ systems than in DLVQ3 (ratio 15:1). The higher MAV is, the lesser variance is obtained.
- DLVQ3 offers at the end of the growing process a solution more redundant than LVQ's. So after pruning, DLVQ3's codebook is smaller than pruned LVQ's.
- We observed that pruned codebook is not completely used for test set. If we pruned again codebook with test set (that is a data set not seen on training) as a reference training error stands. This can be an indication of one of possible roles that should play validation sets in these systems.

MAV			2				10				2	5
Initial Codebook Size			50	150	234	280	50	150	234	280	50	50
EPUP			1	1	1	1	1	1	1	1	Dyn	Dyn
BETA			0.7	0.9	0.9	1.0	0.9	0.5	0.9	0.5	0.7	0.7
RLEN= $(5x10^5)x$			11	4	4	4	4	4	3	4	-	-
No Pruning	Final Codebook Size		494	278	361	723	730	830	723	908	723	908
	Training Error		6.12	6.13	4.9	4.51	4.65	4.17	3.18	3.07	7.04	5.85
	Test Error		9.01	8.48	7.92	7.79	8.01	7.81	7.39	7.23	9.28	8.85
Pruning	Final Codebook Size		351	277	351	413	504	484	656	622	208	242
	Training Error		6.22	6.13	4.9	4.51	4.65	4.17	3.18	3.07	7.04	5.85
	Test Error		9.01	8.48	7.92	7.79	8.01	7.71	7.39	7.23	9.28	8.85
	Class training error	Med.	5.86	5.39	4.93	4.47	4.45	4.18	3.01	2.87	6.61	4.96
		Var.	5.17	5.14	3.45	3.23	1.98	1.37	1.47	1.17	4.35	4.76

Figure 1. Best Simulation Results on experiment 1 (first 8 columns) and experiment 2.

4. Upper and Lower Handwritten Recognition with DLVQ3

In this section, we apply the DLVQ3 for upper and lower handwriting characters recognition using the same partitions of handwritten database and preprocessing introduced in section 3.1. In a first phase we have determined parameter values that allows system to generalize optimally. Then, ten training runs with DLVQ3 were computed using ten different random sequences of training set. Finally, we have proceeded equally on a LVQ system, with codebook of same size, training in cascade with LVQ_PAK programs. Results can be found on figure 4. We can observe that

DLVQ3 generalizes better and is more stable (low test error variance) than LVQ-based system. Also, it has superior performance than other neural approaches reported in [3], where test error on upper & lower recognition are 14.7% and 23.1 % with PCA_{64}+PNN & 10.1% and 20.3% with PCA_{128}+MLP respectively.

Codebook Size		198	245	298	349	412	498	624
Training Error		8.53	8.48	7.55	6.83	6.81	6.25	5.96
Test Error		11.16	11.27	10.66	9.94	10.13	9.67	9.25
Class training error	Med	6.505	6.195	5.83	5.65	5.28	4.69	4.785
	Var	60.64	62.75	56.74	55.7	52.56	53.16	52.49

Figure 2. Simulation results on applying the LVQ_PAK learning strategy to upper handwriting characters (experiment 3).

Expert	Upper				Lower			
	Test Error		Final *C* size		Test Error		Final *C* size	
	Med	Var	Med	Var	Med	Var	Med	Var
PCA_{64}+DLVQ3 (without pruning)	7.27	0.016	893	605.5	15.28	0.021	419	31.15
PCA_{64}+DLVQ3 (with pruning)	7.22	0.015	671	870.8	15.27	0.021	322	62.45
PCA_{64}+LVQ_PAK	9.16	0.074	729	0	17.01	0.843	325	0

Figure 3. Upper & Lower Handwritten Recognition with DLVQ3 & LVQ's algorithm executed in cascade (expressed in table as LVQ_PAK). DLVQ3 parameter values in upper & lower are respectively: initial codebook size=280, EPUP=1, BETA=0.5, MAV=10, RLEN=500000x4; initial codebook size=280, EPUP=1, BETA=0.5, MAV=2, RLEN=500000x4. LVQx's parameter values are the same as section 3.2.4. except olvq1,lvq1,lvq2 *rlen*=190000.

5. Conclusions

A new dynamic LVQ-based classifier has been presented and the obtained results compared with a number of similar strategies. The discussed technique is promising and it will be used as a part of an ICR system.

References

[1] Kohonen, T. "Self-organizing Maps", 2nd Edition, Springer-Verlag (1996)
[2] Kohonen et al. "LVQ_PAK. The Learning Vector Quantization Program Package. Version 3.1", Helsinki University of Technology (1995)
[3] Garris et al. "NIST Form-Based Handprint Recognition System (Release 2.0)", National Institute of Standards and Technology (1997)
[4] Fritzke, B."Growing Self-organizing Networks-Why?", ESANN'96, 61-72 (1996)
[5] Zell et al. "Stuttgart Neural Network Simulator", Version 4.1 (1995)
[6] Fritzke, B. "The LBG-U Method for Vector Quantization-an Improvement over LBG Inspired form Neural Networks", Neural Processing Letters, 5, 35-45 (1997)
[7] Cagnoni, S. & Valli, G. "OSLVQ: a training strategy for optimum-size Learning Vector Quantization classifiers", ICNN'94, 762-765 (1994)