Evolutionary Neural Classification for Evaluation of Retail Stores and Decision Support

Robert Stahlbock
Inst. of Business Information Systems
University of Hamburg
D-20146 Hamburg, Germany
E-mail: stahlboc@econ.uni-hamburg.de

Sven F. Crone
Dep. of Management Science
Lancaster University Management School
Lancaster LA1 4YW, United Kingdom
E-mail: s.crone@lancaster.ac.uk

Abstract— **The neural network paradigm of learning vector quantization (LVQ) and several enhancements of the standard algorithms have demonstrated improved predictive accuracy when applied to simple 'toy' problems. In this paper, we propose a novel approach of evolutionary optimized LVQ classification applied in real-world business decision support. We predict the success of retail outlets of a multinational German company in terms of revenue and profit. The predictions are used to support investment decisions, establishing new stores or closing down existing ones with limited prospective profits. In addition, the predictions provide information to change in-store design or product lines of existing stores. The LVQ networks are trained on data reflecting the macroscopic socio-demographic infrastructure and microscopic in-store aspects of existing outlets. Results of numerous computational experiments in a parallelized PC network are compared with standard neural networks, demonstrating pre-eminent results of the novel method.**

I. Introduction

This paper proposes the application of an evolutionary built artificial neural network on an economic real-world classification problem. A genetic component provides improved selection of input variables for achieving higher classification accuracy. For comparison reasons, manually parameterized standard LVQ is applied as well. Locations of retail stores are classified in terms of sales volume to support decisions that have strong impact on large investments for stores. This main decision has long-term character and induces high fix costs. More detailed decisions on in-store design and assortment of a certain store are more flexible and can more easily be revised in case of an unprofitable decision. All decisions have impact on sales quantity and therefore on the important cash flow.

The organization of this paper is as follows. In Section II, the ideas of standard LVQ are presented briefly. References to enhancements are given. In Section III the new developed combination of evolutionary algorithms and slightly modified LVQ algorithms is shown. Fitness functions are briefly discussed and a parallelized implementation is suggested. The problem – evaluation of locations for retail stores and decision on their in-store design and assortment – is proposed as a multi-class classification task in Section IV. The real-world scenario and computational experiments are described as well as their results. An interpretation of these results is suggested and exemplified. Section V concludes the paper with a summarization and perspectives for further developments.

II. Classification with LVQ

Classification is the process of assigning a class label y to an object with observed measurable attributes expressed in $\vec{x}$. With a sufficient large set of available examples $(\vec{x}, y)$, a machine for supervised learning of the mapping $\vec{x} \rightarrow y$ can be constructed. The objective of a classification process is to find a specific learning machine which captures the relationships in the representative training examples not for building useless overfitted models by memorizing but for generalizing the problem structure from the underlying data generator. The generalization ability allows correct classification of unseen objects based on their attributes' values only. A decision maker should be interested in this performance under generalization conditions.

Learning vector quantization implements a supervised nearest neighbor pattern classifier providing substantial gain in learning speed and performance. The LVQ is regularly applied in pattern recognition, multi-class classification and data compression tasks in widespread areas such as robotics, linguistics or data-mining. In terms of artificial neural networks (ANN), LVQ is a feed-forward, hetero-associative, winner-takes-all ANN, related to self-organizing maps [1]. It includes an input layer (one neuron per input variable), a Kohonen layer with neurons that learn and perform the classification, and an output layer (one node for each class). The number of the hidden Kohonen neurons is either predisposed by the user or dynamically determined by enhanced algorithms.

The weight vector of the weights between all input neurons and a hidden neuron j is called a codebook vector (CV) $\vec{w}_j = (w_{1j}, \ldots, w_{Ij})$. Each CV can be seen as a hidden neuron. A CV is a prototype representing a labelled region (a 'cell') in input space formed by all $\vec{x}$. A class can be represented by an arbitrarily number of CVs, but one CV represents one class only. During training process, the weights are changed in accordance with adapting rules changing the CV's position. The basic LVQ algorithm rewards correct classifications by moving the 'winner' $\vec{w}_w$ (the CV being nearest to the presented input vector $\vec{x}$) towards $\vec{x}$, whereas incorrect classifications are punished by moving the CV in opposite direction. Thus, presented patterns attract prototypes of the correct class, whereas prototypes of other classes are

0-7803-9048-2/05/$20.00 ©2005 IEEE

repelled. Since class boundaries are built piecewise-linearly as perpendicular bisector planes of lines joining pairs of neighboring CVs, the class boundaries are adjusted during the learning process. The tessellation of input space induced by the set of CVs is optimal if all data within a CV's cell indeed belong to the same class. Kohonen has shown that Bayes' optimal class boundaries can be approximated well by LVQ-algorithms.

After the learning process, classification is based on a presented sample's vicinity to the CVs: the classifier assigns a CV's class label to all samples that fall into the CV's cell, i.e. the label of the CV nearest to the sample. The core of the heuristic algorithms, which do not optimize fixed criteria directly, is based on a distance function for comparison between an input vector and all CVs. Usually the Euclidean distance is used. The distance is inversely proportional to the degree of similarity between presented input vector and CVs. The definition of class boundaries by LVQ does not depend on the distance function only, but also on the start positions of CVs, their adjustment rules as well as on the pre-selection of distinctive input features.

The basic LVQ1 suffers from various shortcomings, e.g. the tendency for pushing CVs away from Bayes decision surfaces. Variants have been developed to overcome them and to improve the stability and rate of the convergence process as well as the classification accuracy. Improved versions are e.g. Kohonen's Optimized LVQ1 with faster convergence and LVQ2, LVQ2.1 and LVQ3, that sometimes improve performance. For a comprehensive overview and also details of heuristic learning algorithms of LVQ, readers are referred to standard ANN literature, e.g. [2], [3]. More detailed information can be found in the work of Kohonen, specialized topics are outlined e.g. in [4], [5]. An overview of statistical and neural approaches to pattern classification is given e.g. in [6], [7]. Besides Kohonen's standard algorithms, several extensions from various authors are suggested, e.g. LVQ with conscience [8], LVQ without repulsion [9], Generalized LVQ [10], Learning/Linear Vector Classification [11], Distinction Sensitive LVQ [12], Dynamic LVQ [13], LVQ with Weighted Objective Function [14] or LVQ with Training Count [15]. Newer developments are another Generalized LVQ [16], Generalized Relevance LVQ [17], Robust Soft LVQ [18] or LVQ4-algorithms with promising performance and results [19].

III. Evolutionary development of LVQ

A. Genetic LVQ

Evolutionary algorithms (EA) are meta-heuristics imitating the biological evolution, which can be seen as a long-term optimization process for solving mathematical optimization problems based upon Darwin's 'survival of the fittest'. Problem solutions are abstract 'individuals' in a population. Each solution is evaluated by a fitness function. The fitness value expresses survivability of a solution, i.e. the probability of being a member of the next population and generating 'children' with similar characteristics by handing down genetic information via evolutionary mechanisms like reproduction, variation and selection, respectively. In conjunction with neural nets, EAs can be used for solving problems like, e.g., optimization of the net's weight matrix, the topology or the net parameters, or for determination of the most reasonable input data. Genetic algorithms (GA) are a subgroup of EAs. The main characteristics are a population size greater one and reproduction/variation not only by mutation of gene(s) within a single chromosome but also by the main operator of crossing over. The latter combines genetic information of parents (characteristics of two solutions) in order to get two children (new solutions).

The coding of the problem into a genetic representation, e.g. the sequence of the phenotype's parameters on a genotype, is crucial to the performance of the GA. To be efficient and promising, a GA has to fulfill several criteria, e.g. completeness, compactness and short schemata (see e.g. [20], [21], [22] for an overview and more details).

The EA-based development of ANNs is an iterative process. A net is constructed by transferring the genotype's genetic code into a phenotype. After learning (and cross-validation if appropriate), a net is evaluated by a fitness function. Genetic operations use this quality information for building a new population of nets, which are trained, tested and evaluated again. Thus, the whole learning process can be seen as subdivided into a microscopic cycle for learning of a net and a macroscopic evolutionary one. The iterative learning process of ANNs itself is a part of the whole development process for ANNs, starting from problem modelling and ending at usage and maintenance of a net [23].

Combinations of GAs and LVQ can be found e.g. as G-LVQ in [24] or as LVQ-GA in [25]. The GA of G-LVQ aims at an improvement of number and initial position of hidden neurons for optimizing classification accuracy, net's size and data representation separately. LVQ-GA uses a binary gene representation for input neurons and, therefore, for input data selection, and an integer coding for determination of the number of hidden neurons and their initial positions.

We develop a new GA-LVQ similar to LVQ-GA. The chromosomes contain coding of those net parameters which have to be genetically varied in order to get net variations. In GA-LVQ, the importance or influence of input variables is coded by decimal values controlling 'activity' of input neurons. They affect the algorithm's central measure of distance between CVs and input vectors by weighting components of the Euclidean distance. Moreover, the number of hidden neurons is coded directly. Their initial positions nearby or equal to training patterns are coded by a random seed influencing the process of initializing. Finer parameters like number of learning iterations or learning rate can be coded principally, but are not used in our experiments. The number of output neurons is externally determined by the number of classes. Thus, the decimal coding of input neurons is a refinement of the binary coding. The binary coding allows only binary decisions, whether an input value is used in the classification process – the input neuron is active – or not. The decimal coding does not only include the binary coding, but also allows graduation. This graduation

is implemented by weighting the values of an input feature for calculating the distance, which is a key factor for the results in classification process with a distance based method like LVQ. For each input neuron i (or each input variable, respectively) a decimal weight ψ_i with $0 \leq \psi_i \leq 1$ is coded on a gene. A rough graduation seems to be sufficient, e.g. $\psi_i \in \{0; 0.25; 0.5; 0.75; 1\}$, $\psi_i \in \{0; 0.1; \ldots; 0.9; 1\}$ or even finer. With $\psi_i \in \{0; 1\}$, the decimal coding is equivalent to binary coding. With this 'relevance weight' ψ_i, a modified weighted L_2-norm $\|.\|_2^{\psi}$ can be calculated by $\|\vec{x}\|_2^{\psi} = +\sqrt{\sum_{i=1}^{N} x_i^2 \psi_i}$. A weighted Euclidean distance ϕ_ψ between two vectors $\vec{a}$ and $\vec{b}$ substituting the typically used Euclidean distance ϕ_E is given by using these relevance weights:

$$\phi_\psi(\vec{a}, \vec{b}) = \|\vec{a} - \vec{b}\|_2^{\psi} = +\sqrt{\sum_{i=1}^{N} (a_i - b_i)^2 \psi_i} \; .$$

$\phi_\psi = \phi_E$ is valid for $\psi_i = 1$. The ψ-weighting of a feature i has influence on its impact on the distance by shortening its distance component more (with low factor ψ_i) or less (with high factor ψ_i).

Genetic operators are one- or two-point-crossover and mutation. The crossover is only applied to the gene section coding the input neurons in order to get only genotypes that lead to valid, applicable phenotypes. Furthermore, this section can also be varied by mutation of a single gene, inversion of a gene sequence or exchange of two genes. Genes coding number of hidden neurons and initialization parameter are varied by mutation only, i.e. adding or subtracting a value within a valid range. A high rate for crossing over and low rate for mutation are recommended.

For a high degree of automatization, a destabilization is implemented. It interrupts an optimum search being too local and makes the GA more flexible. Destabilization in this sense means 'death' of accidentally selected 'individuals' and replacement with accidentally constructed new ones. The destabilization occurs if a critical fraction of identical or similar individuals – measured by comparison of all genes or essential gene sections – within a population exists. The number of existing individuals to be taken into the new population can be controlled by a 'rate of survival'. A rate of null is equivalent to a restart of an experiment.

The simplest criterion for selection of parents is the fitness value of a net. Alternatively, a group of some nets differing in their initialization value only can be considered in order to make evaluation more independent from a net's initialization. The group's fitness can be calculated as arithmetic mean of the single values of each net. For a focus more on the impact of a certain initialization value, only the best fitness within a group can be used as selection criterion. Time consuming computations for different initializations can be supported well by parallel implementation of the algorithms (see Sect. III-C). The selection itself can be implemented as tournament selection or as fitness-proportional selection (presuming a reasonable scaling of fitness values). For details of GA-LVQ and the complete ANN model building process see [23].

B. Fitness function

The fitness function is the crucial factor for evaluation and evolution of neural nets providing satisfactory and stable results in real-world applications. In case of fitness maximization, a high fitness value should correspond with good results and a low value with poor results – not only for training data, but also for validation and generalization data. Therefore, it should be checked whether the chosen fitness function favors nets with satisfactory generalization ability in order to select useful neural nets systematically instead of accidentally. For a representative overview, it is helpful to compute and evaluate as many nets as possible in maintainable time (hundreds/thousands). Again, parallelized implementation helps in performing the necessary high computing power.

The fitness function should represent the user's objective. In conjunction with ANN, it allows to evaluate and control the superordinated evolutionary learning process, thus enhancing the goals that the nets' algorithms are aiming at. For some classification tasks, the mean classification rate (MCR) may be sufficient. Taking (asymmetric) costs for misclassifications into account seems to be more versatile and also realistic (see e.g. [26] with application of asymmetric cost functions for time series prediction). A fitness functions F for three-class-problems can be e.g. one of the following exemplary ones (with n_y^+ as number of correctly classified pattern of class y, m_y as number of all patterns belonging to class y and $CR_y = n_y^+/m_y$), applied to a training subset (or better a hold-out subset) of data:

$$F = MCR = (CR_1 + CR_2 + CR_3)/3 \tag{1}$$

$$F = CR_1 \cdot CR_3 \tag{2}$$

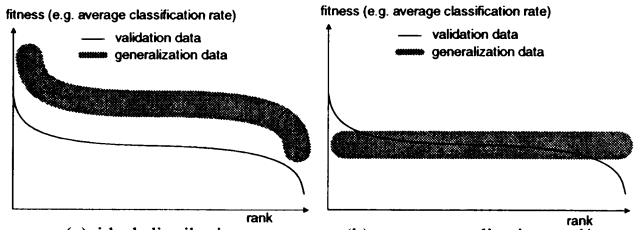
$$F = CR_1 \cdot CR_2 \cdot CR_3 \tag{3}$$

$$F = MCR^{\text{learn}} \cdot MCR^{\text{validation}} \tag{4}$$

Fitness function (2) aims at uniform high classification rates for classes 1 and 3, whereas fitness (3) favors consistent classification rates for all three classes. Functions aiming at even results on two different subsets of data can be constructed too, see e.g. fitness (4). Instead of searching for an all-purpose net for all classes, an approach for finding 'expert nets' specialized for the subtask of recognize patterns belonging to a special class may be sufficient. Besides these direct measures of performance in terms of classification rates, a fitness function can include 'costs' for net complexity, e.g. expressed as number of input variables and/or hidden neurons (as in [25]). In general, less complexity means better generalization ability. Thus, the formulation of an appropriate fitness function is highly dependent on the specific problem to be solved.

Capability of both the LVQ nets and the used fitness function can be evaluated graphically. The results of nets are sorted in descending order of the fitness value (in case of maximizing fitness). The separate charting of fitness values or classifications rates for different data sets (used for learning and tests of validation and generalization) in a two-dimensional scatter diagram allows for comparison of values

per x-coordinate and evaluation of the degree of correlation between the fitness used to control the evolutionary process and the fitness for unknown ('real') data. Ideally, results on training/validation/generalization data are rudimentary proportional (see s-curve in Fig. 1(a)). It can be assumed that a net with high fitness value also yields a satisfactory result for unknown data and therefore can be reasonably selected for application. Fig. 1(b) shows the contrast: it is impossible to conclude that a high fitness value implies a good generalization result. Nets with good results in application are not preferred systematically within the evolutionary process. A selected net is more or less arbitrarily 'good' (or 'poor'). Note, that identical scaling on y-axis for both plots is not necessary.



(a) ideal distributions

(b) poor generalization and/or improper fitness function

Fig. 1. Fitness (validation), corresponding results (generalization)

C. Parallelized Implementation

Parallelization allows higher computational performance, i.e. a higher number of calculated nets per time unit, which is nearly proportional to the number of clients in a PC network. Overhead of technical management and communication is negligible. More calculated nets mean more results, that allow a more valid evaluation and analysis of the solution methods' quality. Among other forms of parallelization (see e.g. [13]), at least the following three ones (or combinations of them) can be applied in combination with LVQ:

- Initialization: one LVQ with m different initializations is calculated on m computers. This allows a more general evaluation of a net topology independent of the initialization, which may have strong impact on results.
- Data: m different combinations of data sets are used on m computers. A better estimation of generalization ability seems to be provided.
- Individuals: m LVQs of a population (in case of GA) are computed simultaneously on m computers.

The genetic development of LVQs can be parallelized in a scalable PC network with one server administrating the population and the genetic algorithm, and one to m LVQ-clients, each of them getting instructions from the server for computing one LVQ at a time and delivering the result, i.e. a fitness value.

We develop server and client software, coded in Visual Basic and C. Minimum equipment is one PC running both server and client. The maximum reasonable number of clients is the number of individuals in a population. A more efficient number is in the range of the average number of solutions to be calculated for a new population. At the moment, GA-LVQ supports Kohonen's LVQ1 and LVQ2.1 only, but enhanced LVQ algorithms can be implemented easily. Verifying experiments succeed: GA-LVQ is applied to the well known iris dataset with additional useless input variables, that are correctly recognized and eliminated by ψ-weights $= 0$.

IV. Empirical Evaluation

A. Evaluation of retail stores

For the evaluation of a location of an existing store in terms of sales volume or of an eligible location of a newly planned store, a more or less rough classification of (expected) sales volume is sufficient (for theory and details of locations' evaluation see e.g. [23]). Therefore, forecasting or estimating the sales volume can be seen as a classification problem. Results of a classification process should be used as completion of knowledge of human experts working in the field, e.g. for building a priority list for locations to be inspected in detail, not as substitution for invaluable human skills.

The data pool consists of external macroscopic up-to-date data describing socio-demographic, economical infrastructure at a specific location, e.g. number of homes and residents, retail turnover for different branches of trade, discretionary buying power of a region or number and type of retailers, as well as internal microscopic data describing economical and technical figures of existing stores, e.g. kind of assortment, configuration/equipment components, year of last redecoration, sales volumes for different periods, or sales area. A total of 26 input variables is given. Sales volumes of the product line we are interested in are partitioned in three classes corresponding to three possible decisions concerning location policy:

- high: 'establish new store/continue business at store',
- medium: 'analyze location more detailed, e.g. assortment or in-store design,
- low: 'do not establish new store/shut down existing store'.

The task is to classify locations/stores, that are described in numerical data patterns of values for external and internal properties of locations and stores. Furthermore, interpretation of classification results should lead to decisions on assortment and in-store design for upgrading a store to be a member of a class with higher sales volume in the future.

B. Computational experiment in a real-world scenario

For each class approximately the same number of examples is available. Sales volume of stores that are opened or closed within a calendar year are extrapolated based on a seasonal (monthly) distribution. Furthermore, data are pre-partitioned for large, medium and small cities. Class memberships differ for each type of city, because decision rules and definitions of 'high', 'medium' or 'low' sales volumes differ. Disjoint data sets for learning, validation and generalization tests are randomly selected from the entire data. Different constellations are used for several experiments in order to get general results for estimation of the methods' performance. Learning data consist roughly of 40-70 % of available data records,

validation data has 5-25 %, the rest forms the hold-out-set for out-of-sample evaluation of the classifier's generalization performance. For example, 1 250 data records are available for large cities. They are separated in 940 (840, 740) records forming the learning set, 220 (270, 320) forming the validation set and 90 (140, 190) forming the generalization set.

Standard LVQ and basic extensions (e.g. turned off repulsion in the early training phase or usage of conscience) are computed as implemented within the commercial software package NeuralWorks Professional II/Plus with a single PC providing benchmark results. For these manually developed nets, an iterative heuristic approach to determine appropriate net architectures (e.g. number of hidden neurons, learning rate) is chosen . Each network is randomly initialized with 4 different random seeds leading to alternative (but reproducible) starting positions of CVs. The number of CVs is set to 5%, 10%, ...25% of the number of training examples (approximately; each class is represented by same number of CVs). A standard early stopping rule with variation of its parameters is used in order to avoid overfitting from learning data. Alternatively, variations in fixed numbers of iterations (10/100/500 times the number of learning examples) are analyzed. All 26 input variables are taken into account. Learning schedules, i.e. sequences of Kohonen's standard algorithms, scaling of input variables and other parameters are taken with standard adjustments recommended in the documentation of NeuralWorks or pre-set by the program unless otherwise noted.

In case of GA-LVQ, computational experiments are performed on a network of Pentium-PCs with one server and up to 60 clients as clients. In most experiments only 20 to 30 clients are used simultaneously in an experiment. Various fitness functions based on classification rates and including penalty costs proportional to complexity are applied to the validation data. Both an early stopping rule and fixed numbers of iterations are tested as well as variants with binary genetic codes for active/inactive input neurons and decimal codes for gradual activity and resulting weighted input values. As a result of pre-tests, a population size of 200 seems to be reasonable. An elitist selection, that always carries over the best net into the next population, is used.

C. Results, analysis and interpretation

Generally, the results of GA-LVQ dominate the results of manually adjusted LVQs. Within of GA-LVQ results show no dominant fitness function. All classification rates, especially the MCR on generalization data, are in the same satisfying order of magnitude for all three types of city (see Table I, showing results from those nets that are selected for generalization tests due to their promising results on training and validation data).

Results of the top 10 % of the nets (in respect of MCR on validation data) show only a slight higher minimum MCR on generalization data for the manually adjusted LVQs (up to seven basis points). Even those results are not in the range of the GA-LVQ's results. As expected, the fitness charts of GA-LVQ never show an ideal s-curve for hold-out data, but

TABLE I

MCR FOR GENERALIZATION DATA (MIN/MAX, GROUP OF 'BEST' NETS)

Type of city	Small		Medium		Large	
MCR[%]	min	max	min	max	min	max
LVQ conscience & LVQ1 & LVQ2	57.7	71.1	50.1	60.7	54.2	58.8
LVQ consc. & no repuls. & LVQ2	58.4	71.6	50.1	64.1	57.2	59.4
GA-LVQ	68.6	69.7	69.9	71.7	71.8	70.0

a mixture between Fig. 1(a) and Fig. 1(b) with a slightly decreasing fitness and more variance from high to low rank. On the other hand, charts of classification rates of conventionally developed nets clearly tend to be similar to Fig. 1(b) with an almost horizontal scatter plot for generalization data. The classification rates on validation data give less or no information about nets' quality. The user is not able to select a net that generalizes well and reliably.

Between 7 and 16 input variables are weighted with 0, the half of the rest is weighted with 0.5. Finer codings have no observable impact on results. The number of hidden neurons depends on the number of used variables and is in the wide range of 30 to 108. Using the non-zero-weighted inputs as fully weighted inputs for NeuralWorks shows inferior or similar results, but no improvement at all.

A classification result supports a decision on a location directly. Furthermore, it allows answering the following interesting questions:

- What are the main properties of a store that make it a member of a special class? Which decisions on in-store design influence sales volume?
- Which properties of a location are responsible for a store's class of sales volume?
- Which potential for development does a store have? How can this potential be used advantageously?

The basic idea is to take advantage of a distance based classifier and interpret CVs and distances between them and input pattern(s). Distance components (weights w_{ij} from input neuron i to hidden neuron j), that have similar values for all hidden neurons, result only in small distance differences. Therefore, the corresponding input features have no strong impact on the classification result. The idea is exemplified assuming evenly relevant weights of inputs, use of Euclidean distance and only one CV per class.

Some typical properties of a store are given as 'format', 'type', 'especially equipped' and 'duration'. Exemplified weights for analysis of are presented in Table II. Data show strong influence of 'type'. A store with a type coded as 1 will be rather classified belonging to class 1 (high sales volume), a type 0 (scaled to -1) to class 3 (low sales volume). Special equipment (not specified here) or a short duration (time period from opening or redecoration) seems to stimulate sales. The 'format' has less influence. Answers for above mentioned questions can be derived from similar considerations and what-if-analyses by varying certain values and computing distances for classifying. These analyses and interpretations are not bound to usage of the proposed evolutionary LVQ version. The main results of the GA-LVQ experiments are feasible:

TABLE II

WEIGHTS OF LVQ WITH THREE CVS FOR THREE CLASSES (CUT-OUT OF EXEMPLIFIED DATA, SCALED INTO $[-1; 1]$)

Property	Class 1 w_{i1}	Class 2 w_{i2}	Class 3 w_{i3}	Weight span
Type	0.65	0.10	−0.42	1.07
Special equipment	0.41	0.25	−0.67	1.08
Format A	−0.97	−0.81	−0.63	0.34
Format B	−0.68	−0.88	−0.98	0.30
Format C	−0.74	−0.71	−0.85	0.14
Duration	−0.32	−0.09	0.07	0.39

they meet with approval of human experts involved in the topic of retail stores and evaluation of locations. It should be noted that classifiers should be seen as add-on tools for decision support besides profound expert knowledge.

V. CONCLUSIONS

The proposed GA-LVQ shows promising results for decision support in a complex economic real-world classification problem. The results dominate results of Kohonen's conventionally applied standard LVQs. A fitness function allows control in respect of decision maker's goal beyond the potentials of the LVQ-inherent heuristics. Common problems like reasonable initialization and values for learning parameters like e.g. number of learning steps remain, but they are mitigated by computation of a high number of nets not with brute force but rather with 'evolutionary intelligence' in a parallel PC network implementation. Within the model building process, decisions on ANN's topology are automated. On the one hand the degrees of freedom concerning parameters of neural learning are reduced, on the other hand new ones concerning evolutionary learning arise. The shown method may be enhanced for improvement of results. Enhancements can be e.g. usage of more LVQs in an 'expert committee' (each LVQ 'votes' for a class; influence of different voting rules can be studied), combination of different classification methods (e.g. support vector machines; again in a committee), rejection of classification (if distance between winning CV and input pattern is too high) or consideration of knowledge of a-priori probabilities. Furthermore, the examination of the influence of different fitness functions on classification results is an interesting research topic. An important information for decision makers is the quality or significance of results. This information can possibly be provided by interpretation of the distances between CVs and the object to be evaluated. Further studies may also focus on comparison between GA-LVQ and newer LVQ enhancements like LVQ4 or on implementation of these newer LVQ algorithms into GA-LVQ. Moreover, the GA-LVQ should be applied to other real-world classification problems in order to get a broader overlook and better evaluation of the method's overall performance.

REFERENCES

[1] T. Kohonen, *Self-Organizing Maps*, 2nd ed. Berlin: Springer, 1997.

[2] L. Fausett, *Fundamentals of Neural Networks : architectures, algorithms, and applications*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[3] D. W. Patterson, *Artificial neural networks: theory and applications*. Singapur: Prentice Hall, 1996.

[4] J. T. Laaksonen, "A method for analyzing decision regions in Learning Vector Quantization algorithms," in *Artificial Neural Networks, 2 – Proc. ICANN-92, Proceedings of International Conference on Artificial Neural Networks (Brighton, GB)*, I. Aleksander and J. Taylor, Eds., vol. 2. Amsterdam: Elsevier, 1992, pp. 1181–1184.

[5] N. Kitajima, "A New Method for Initializing Reference Vectors in LVQ," in *Proc. ICNN '95, Proceedings of IEEE International Conference on Neural Networks (Perth, Australia)*, vol. 5. IEEE Service Center, 1995, pp. 2775–2779.

[6] J. Schürmann, *Pattern classification: a unified view of statistical and neural approaches*. New York: Wiley & Sons, 1996.

[7] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. New York: Springer, 2001.

[8] D. DeSieno, "Adding a Conscience to Competitive Learning," in *Proc. ICNN '88, Proc. of IEEE Int. Conf. on Neural Networks*, vol. I. Piscataway, NJ: IEEE Service Center, 1988, pp. 117–124.

[9] NeuralWare , Inc., *Neural Computing : A Technology Handbook for Professional II/Plus and NeuralWorks Explorer*, NeuralWare, Inc., Technical Publications Group, Pittsburgh, PA, 1993.

[10] N. R. Pal, J. C. Bedzek, and E. C.-K. Taso, "Generalized Clustering Networks and Kohonen's Self-Organizing Scheme," in *IEEE Transactions on Neural Networks*, vol. 3, no. 4, 1993, pp. 546–557.

[11] M. Verleysen, P. Thissen, and J.-D. Legat, "Linear vector classification: An improvement on LVQ algorithms to create classes of patterns," in *New Trends in Neural Computation – Proc. of Int. Workshop on ANN (IWANN '93), Sitges, Spain, June 1993*, J. Mira, J. Cabestany, and A. Prieto, Eds. Berlin: Springer, 1993, pp. 340–345.

[12] M. Pregenzer, D. Flotzinger, and G. Pfurtscheller, "Distinction Sensitive Learning Vector Quantisation – a new noise-insensitive classification method," in *Proc. of Int. Conf. on Neural Networks (ICNN '94), Orlando, FL*, vol. V. Piscataway, NJ: IEEE Service Center, 1994, pp. 2890–2894.

[13] A. Zell, *Simulation Neuronaler Netze*. Bonn: Addison-Wesley, 1994.

[14] S.-J. You and C.-H. Choi, "LVQ with a Weighted Objective Function," in *Proc. ICNN '95, Proc. of IEEE Int. Conf. on Neural Networks (Perth, Australia)*, vol. 5. IEEE Service Center, 1995, pp. 2763–2768.

[15] R. Odorico, "Learning Vector Quantization with Training Count (LVQTC)," *Neural Networks*, vol. 10, no. 6, pp. 1083–1088, 1997.

[16] A. Sato, "An analysis of initial state dependence in generalized lvq," in *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 928–933.

[17] M. Strickert, "Self-organizing neural networks for sequence processing," Ph.D. dissertation, Dep. of Mathematics and Computer Science, Univ. of Osnabrück, Germany, 2004.

[18] S. Seo and K. Obermayer, "Soft Learning Vector Quantization," *Neural Computation*, vol. 15, pp. 1589–1604, 2003.

[19] M.-T. Vakil-Baghmisheh and N. Pavesic, "Premature clustering phenomenon and new training algorithms for LVQ," *Pattern Recognition*, vol. 36, no. 8, pp. 1901–1912, 2003.

[20] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading (Mass.): Addison-Wesley, 1989.

[21] J. H. Holland, *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge, MA: MIT Press, 1994.

[22] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin: Springer, 1994.

[23] R. Stahlbock, *Evolutionäre Entwicklung künstlicher neuronaler Netze zur Lösung betriebswirtschaftlicher Klassifikationsprobleme*. Berlin: WiKu, 2002.

[24] J. Merelo and A. Prieto, "G-LVQ, a combination of genetic algorithms and LVQ," in *Artificial Neural Nets and Genetic Algorithms – Proc. of the Int. Conf. 1995 (Alès, France)*, D. W. Pearson, N. C. Steele, and R. F. Albrecht, Eds. Wien: Springer, 1995, pp. 92–95.

[25] U. Derigs and G. Schirp, "Genetische Modellierung von Künstlichen Neuronalen Netzen : Erfahrungen beim Einsatz zur Kreditwürdigkeitsprüfung," *OR Spektrum*, no. 19, pp. 285–293, 1997.

[26] S. F. Crone, "Training Artificial Neural Networks using Asymmetric Cost Functions," in *Proc. 9th Int. Conf. on Neural Information Processing (ICONIP'02) – Computational Intelligence for the E-Age, Nov 18-22, 2002, Singapore*, L. Wang, C. Rajapakse, K. Fukushima, S.-Y. Lee, and X. Yao, Eds., vol. 5. Piscataway, NJ: IEEE Service Center, 2002, pp. 2374–2380.