SUPPORT VECTOR MACHINES VERSUS ARTIFICIAL NEURAL NETWORKS – NEW POTENTIAL IN DATA MINING FOR CUSTOMER RELATIONSHIP MANAGEMENT?

Sven F. Crone ; Stefan Lessmann ; Robert Stahlbock

Abstract. In competitive consumer markets, data mining for customer relationship management faces the challenge of systematic knowledge discovery in large data streams to achieve operational, tactical and strategic competitive advantages. Methods from computational intelligence, most prominently artificial neural networks and support vector machines, compete with established statistical methods in the domain of classification tasks. As both methods allow extensive degrees of freedom in the model building process, we analyse their comparative performance and sensitivity towards data pre-processing in real-world data. In addition to simpler configuration, support vector machines robustly outperformed various neural network paradigms in classification. Consequently, they are recommended as a contemporary method for data mining in analytical customer relationship management.

Keywords: Customer Relationship Management, Data Mining, Learning Vector Quantisation, Multilayer Perceptron, Support Vector Machines

1 INTRODUCTION

The customers of a company are regarded as valuable business resources in competitive markets, leading to efforts to systematically prolong and exploit existing customer relations. Therefore, the strategies and techniques of customer relationship management (CRM) required to gain knowledge about customer behaviour and preferences through data mining of large databases has received increasing attention in management science.

In data mining, conventional statistical methods routinely applied to classification tasks suffer certain drawbacks in their inability to capture non-linear coherences in addition to requiring a priori assumption for the model building process. Recently, various paradigms of artificial neural networks (ANN) and support vector machines (SVM) capable of solving classification problems in similar domains, have found consideration in practice, promising effective and efficient solutions for managerial classification problems in real-world applications. However, successful applications to CRM related tasks are still limited. Additionally, both classes of soft computing methods allow severe degrees of freedom in the model-building process through extensive parameters. In addition, different variations of data pre-processing through scaling, encoding etc. raise degrees of freedom prior to the actual data mining phase even further.

In empirical business decisions, performance may not only be based on selecting the most effective method, achieving the lowest classification error, but also the most efficient and robust method, balancing the potential trade-off between costs, time and quality in order to derive a solution considered pre-eminent in real world scenarios.

Subsequently, we conduct an experimental evaluation of the competing methods in the domain of aCRM, striving to exemplify the adequacy and performance of ANN versus SVM for the task of response optimisation based upon an empirical, numerical experiment from an ongoing project with a large publishing house.

Following a brief introduction to data mining within CRM, section 3 assesses the competing approaches of different ANN paradigms and SVMs in classification tasks, highlighting the degrees of freedom in the modelling process. This is followed by an experimental evaluation of their competitive performance on an empirical dataset in section 4. Conclusions are given in section 5.

2 DATA MINING IN CUSTOMER RELATIONSHIP MANAGEMENT

2.1 Motivation and Process of Knowledge Discovery in Databases

In an increasingly competitive market, caused by inconsistent consumer behaviour, escalating globalisation and the extending possibilities to conduct business over the internet in a recessive global economy, the customers of a company are regarded as key business resources [26]. Consequently, (aCRM) has received increasing attention in management science as a systematic approach to strategically prolong and exploit these valuable customer relations, providing the tools and infrastructure to record and analyze customer centred information in order to build up longer lasting and more profitable customer relationships [3, 10]. The analytical process of collecting, assembling and understanding the profound knowledge about customer behaviour and preferences in aCRM is referred to as knowledge discovery in databases (KDD).

The process of KDD (see Fig. 1), is initiated by formulating an economical goal as a prerequisite, building the general framework of analytical techniques. Following the phase of data selection, where utilisable data sets are evaluated, catalogued and combined with external data to extract a raw data set for further processing and to achieve data understanding, the following phase of data pre-processing and cleaning is most important and crucial to the quality of the KDD process and the final results. Identification and removal of missing values, deciding on strategies for handling noisy data, accounting for time-sequence information and correlation are subject to this stage, as well as the selection of a suitable sampling strategy to consider computational cost, memory requirements, accuracy of estimator, and the general sampling approach (random, stratified, under-/over-/ average sampling) to derive an efficient sample size [20, 29, 49], as the use of all available data may prove to be computationally inefficient or even prohibitive for real world problems. Subsequently, data transformations are required to ensure a mathematical feasible data format for the proceeding application of a specific data mining algorithm, mainly encoding attributes of nominal or ordinal scale, standardizing numerical data to predefined intervals or reduction of dimensionality and feature construction through principal component analyses etc.

Utilising the processed and transformed data set, the stage of data mining consist of selecting and applying a suitable data mining method [27, 28] in order to identify hidden patterns in the data relevant to business decisions [7] through a partially automated analysis. Combining a robust model with contemporary relational database systems and the emergent culture of data integration, data mining promised to capitalize on customer knowledge formerly buried within the information systems [67]. The results must be evaluated not only regarding precision and statistical

significance but also economical relevance. A more detailed discussion about KDD and different stages can be found in [14].

Although the activities described above follow a linear sequence, KDD usually follows a highly recursive structure, typically partitioning the data set into a training sample, used to parameterize the data mining method, and a hold-out or generalisation set, enabling evaluation of the results obtained on the basis of unseen data. The usage of data never entering the training or model building stage for evaluation is crucial, as we are ultimately interested in discovering patterns, which are generally applicable. If the evaluation reveals insufficient generalisation performance, it's common practice to go back to a previous stage in the KDD process, trying to improve by changing for example the applied data mining method, sampling strategy or any other component of the overall process.

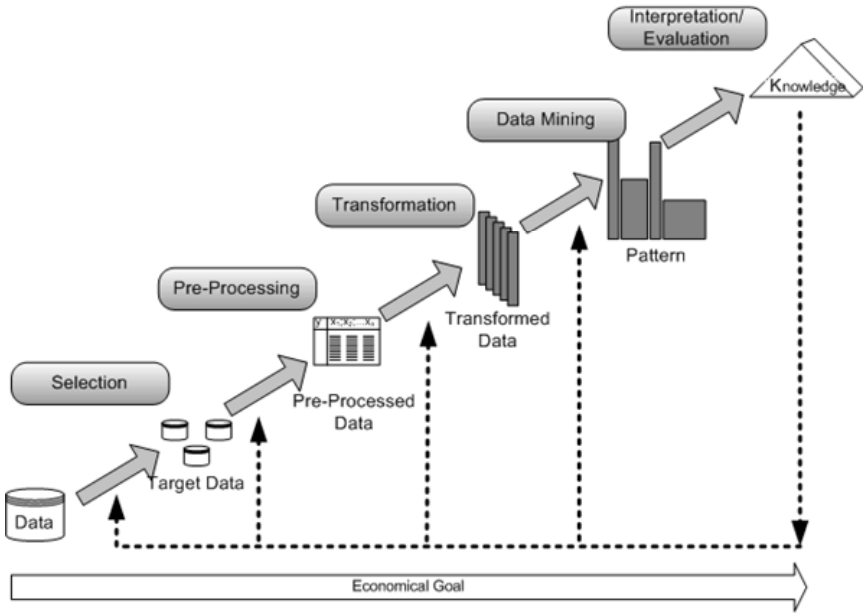

Fig. 1: Data mining in the process of knowledge discovery in databases [14]

2.2 Data Mining in analytical customer relationship management

Data mining activities may be distinguished in distinct processes categories [39]. While discovery focuses on searching a database for hidden patterns without a predefined hypothesis about the nature of the pattern and deriving a model of the causal generator of the data, predictive modelling aims to predict future behaviour based upon the model previously discovered in patterns. Forensic Analysis applies the extracted model or patterns to find anomalies in the data. Generally, a model of a structural dependency derived from data may be used for explanation, differentiation of extrapolation into the future.

The data mining objective defines the use of a segmentation, association; regression or classification model; see [24] for an elaborate discussion. Data mining problems in the aCRM domain, such as response optimisation to distinguish between customers who will react to a mailing campaign or not, churn prediction, in the form of classifying customers for churn probability, cross-selling, or up-selling are routinely modelled as classification tasks, predicting a discrete, often binary feature using empirical, customer centred data of past sales, amount of purchases, demographic or psychographic data etc.

Conventional statistical methods of logistic regression, discriminant analysis or decision trees are routinely applied to data mining. However, most conventional statistical methods suffer certain drawbacks in real-world scenarios due to their inability to capture nonlinear coherences in addition to requiring a priori assumption for the model building process. Recently, various architectures from computational intelligence and machine learning, such as artificial neural networks (ANN) and support vector machines (SVM) have found increasing consideration in practice, promising effective and efficient solutions for managerial classification problems in real-world applications through robust generalisation in linear and non-linear classification problems, deriving relationships directly from the presented sample data without prior modelling assumptions.

Following, we will give a brief discussion on the different classification approaches of the competing soft computing methods

3 COMPETING SOFT COMPUTING METHODS FOR CLASSIFICATION

3.1 Learning Machines for Classification

Data driven methods from computational intelligence share the common approach of learning machines in classification for data mining [17]. Let all relevant and measurable attributes of an object, e.g. a customer, be combined in a vector $\boldsymbol{x}$ and the set $X = \{\boldsymbol{x}_i, ..., \boldsymbol{x}_n\}$ denotes the input space with n objects. Each object belongs to a discrete class $y \in Y$ and we will refer to a pair $(\boldsymbol{x}, y)$ as an example of our classification problem. Presuming that it is impossible to model the relationship between attribute vector $\boldsymbol{x}$ and class membership y directly, either because it is unknown, to complex or the data is corrupted by noise, and that a sufficient large set of examples $S = ((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_l, y_l)) \subseteq (X \times Y)^l$ is available, we can incorporate a machine to learn the mapping between $\boldsymbol{x}$ and y. The learning machine is actually defined by a set of possible mappings $x \to f(\boldsymbol{x}, \boldsymbol{\alpha})$, where the functions $f(\boldsymbol{x}, \boldsymbol{\alpha})$ themselves are labeled by the adjustable parameter vector $\boldsymbol{\alpha}$ [7]. The objective is to modify the free parameters $\boldsymbol{\alpha}$ to find a specific learning machine which captures the relationships in the training examples, $f_{\boldsymbol{\alpha}}(\boldsymbol{x}_i) \approx y_i \ \forall i = (1, ..., i)$, incrementally minimizing a given objective function and generalizing the problem structure within to allow correct estimation of unseen objects on the basis of their attribute values x_i.

For most questions of parameterisation only rules of thumb are known. Answers that are valid for all kinds of problems cannot be given – each problem needs its own. Therefore, knowledge not only in the field of soft computing but also in the problem domain is necessary. Following, we outline the specific modelling-properties for classification for alternative network paradigms. For a comprehensive discussion readers are referred to [4, 8, 17, 36, 38]

3.2 Multilayer Perceptrons

Multilayer perceptrons (MLPs) represent the most prominent and well researched class of ANNs in classification, implementing a feedforward, supervised and hetero-associative paradigm [4, 36, 51]. MLPs consist of several layers of nodes u_j, interconnected through weighted acyclic arcs w_{ij} from each preceding layer to the

following, without lateral or feedback connections [4, 31, 32, 36].[1] Each node calculates a transformed weighted linear combination of its inputs of the form $f_{act}\left(\boldsymbol{w}^T\boldsymbol{o}\right)$, with $\boldsymbol{o}$ the vector of output activations o_j from the preceding layer, $\boldsymbol{w}^T$ the transposed column vector of weights w_{ij}, and f_{act} a bounded non-decreasing non-linear function, such as the linear threshold or the sigmoid, with one of the weights w_{0j} acting as a trainable bias θ_j connected to a constant input $o_o = 1$ [36]. Fig. 2 gives an example of a MLP with a [3-4-1] topology:

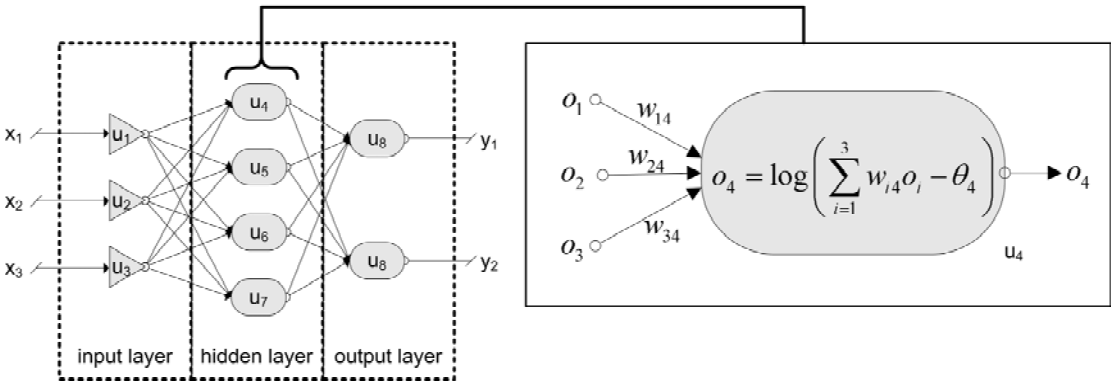

Fig. 2: Three layered MLP showing the information processing within a node, using a weighted sum as input function, the logistic function as sigmoid activation function and an identity output function.

For pattern classification, MLPs adapt the free parameter $\boldsymbol{w}$ through supervised training to partition the input space through linear hyperplanes. To separate distinct classes, MLPs approximate a function of the form $g(\boldsymbol{x}): X \to Y$ which partitions the X space into polyhedral sets or regions, each one being assigned to one out of the m classes of Y. Each node has an associated hyperplane to partition the input space into two half-spaces. The combination of the individual, linear node-hyperplanes in additional layers allows a stepwise separation of complex regions in the input space, generating a decision boundary to separate the different classes [36]. The orientation of the node hyperplanes is determined by the relative sizes of w_{ij} in $\boldsymbol{w}$ including the threshold θ_j of a node u_j, modelled as an adjustable weights w_{0j} to all nodes u_j to offset the node hyperplane along $\boldsymbol{w}$ for a distance $d = \theta_j \left\|\boldsymbol{w}\right\|$ from the origin to allow flexible separation. [30] The node non-linearity f_{act} determines the output change as the distance from x to the node hyperplane. In comparison to threshold activation functions with a hard-limiting, binary class border, the hyperplanes associated with sigmoid nodes implement a smooth transition from 0 to 1 for the separation [23] allowing a graded response depending on the slope of the sigmoid function and the size of the weights. The desired output as a binary class membership is often coded with one output node $y_i = \{0;1\}$ or for multiple classification n nodes with $y_i = \{(0,1);(1,0)\}$ respectively [23].

[1] We count all L layers of a network, with L-1 active, computing layers excluding the input layer, instead of the layers of trainable weights due to the possibility of shortcut connections etc.

The representational capabilities of a MLP are determined by the range of mappings it may implement through weight variation. [36] Single layer perceptrons are capable of solving only linearly separable problems, correctly classifying data sets where the classes may be separated by one hyperplane [36]. MLPs with three layers are capable to approximate any desired bounded continuous function. The units in the first hidden layer generate hyperplanes to divide the input space in half-spaces. Units in the second hidden layer form convex regions as intersections of these hyperplanes. Output units form unisons of the convex regions into arbitrarily shaped, convex, non-convex or disjoint regions [1, 45]. Fig. 3 exemplifies this in a [2-6-2-1] network.

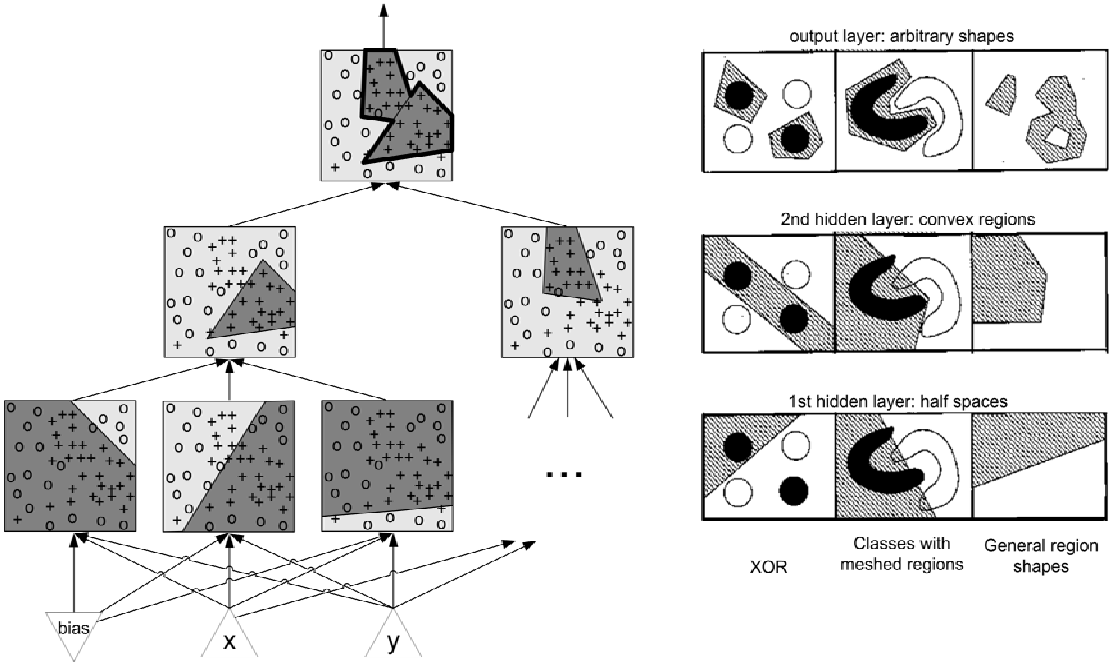

Fig. 3: Partitioning of the input space by linear threshold-nodes in an ANN with two hidden layers and one output node in the output layer and examples of separable decision regions [9]. For sigmoid activation functions smooth transitions instead of hard lined decision boundaries would be formed.

Given a sufficient number of hidden units, a MLP can approximate any complex decision boundary to divide the input space with arbitrary accuracy, producing a (0) when the input is in one region and an output of (1) in the other [28]. This property, known as a universal approximation capability, poses the essential problems of adequate model complexity in depth and size, i.e. the number of nodes and layers, and controlling the network training process to prevent overfitting. As perfect classification on training data does not necessitate generalisation for optimal separation of previously unseen data, simpler models with fewer parameters and training using early-stopping with out-of-sample evaluation on separate datasets are generally preferred.

The network paradigm of MLP offers extensive degrees of freedom in modelling for classification tasks. Structuring the degrees of freedom, each expert must decide upon the static architectural properties P, the signal processing within nodes U, learning algorithm L and the pre-processed datasets D [15] in order to achieve the design goal, characterised through the objective function or error function O [22], calling for decisions upon $ANN=[P, L, U, D, O]$. The topology of the net is determined through the size N^S and depth N^L, of the network (number of layers,

number of nodes in each hidden layer and coding of output vector through nodes in the output layer[2]), connectivity of the weight matrix K (fully or sparsely connected, shortcut connections etc.) and the activation strategy T (feedforward or with feedback): $P=[N^S, N^L, K, T]$. The signal processing within nodes, is determined by input function S (weighted sum or product, distance measures etc.), activation function A (tanh, logistic, sin, etc. with offsets, limits etc.) and output function F (linear, winner takes all variants or softmax), leading to $U=[S,A,F]$. Decisions concerning the learning algorithm encompass the choice of learning algorithm G (backpropagation, one of its derivatives, higher order methods or heuristics etc.), the complete vector of learning parameters for each individual layer and different phases in the learning process P^{TL}, the procedure I^P and number of initialisations for each network I^N and the choice of the stopping method for the selection of the best network solution B. For classification, minimizing a squared error measure as the objective function O is inapplicable, as the goal is maximisation of correct classification. Consequently, the specification requires decisions upon $MLP=[[N^S, N^L, K, T], [S, A, F], [G, P^T, I^P, I^N, B], D, O]$, with the question of data pre-processing pre-determined for all competing methods in an early step of the knowledge discovery process.

3.3 Learning Vector Quantisation

Learning Vector Quantisation (LVQ) is a supervised version of vector quantisation, similar to Selforganising Maps (SOM) based on work of LINDE et al. [4, 13, 18], GRAY [33] and KOHONEN (see [21, 25] for a comprehensive overview). It can be applied to pattern recognition, multi-class classification and data compression tasks, e.g. speech recognition, image processing or customer classification. As supervised method, LVQ uses known target output classifications for each input pattern of the form $(\boldsymbol{x}, y)$.

LVQ algorithms do not approximate density functions of class samples like Vector Quantisation or Probabilistic Neural Networks do, but directly define class boundaries based on prototypes, a nearest-neighbour rule and a winner-takes-it-all paradigm. The main idea is to cover the input space of samples with 'codebook vectors' (CVs), each representing a region labelled with a class. A CV can be seen as a prototype of a class member, localized in the centre of a class or decision region ('Voronoï cell') in the input space. As a result, the space is partitioned by a 'Voronoï net' of hyperplanes perpendicular to the linking line of two CVs (mid-planes of the lines forming the 'Delaunay net'; see Fig. 4). A class can be represented by an arbitrarily number of CVs, but one CV represents one class only.

---

[2] The number of input nodes is pre-determined through data coding and variable selection.

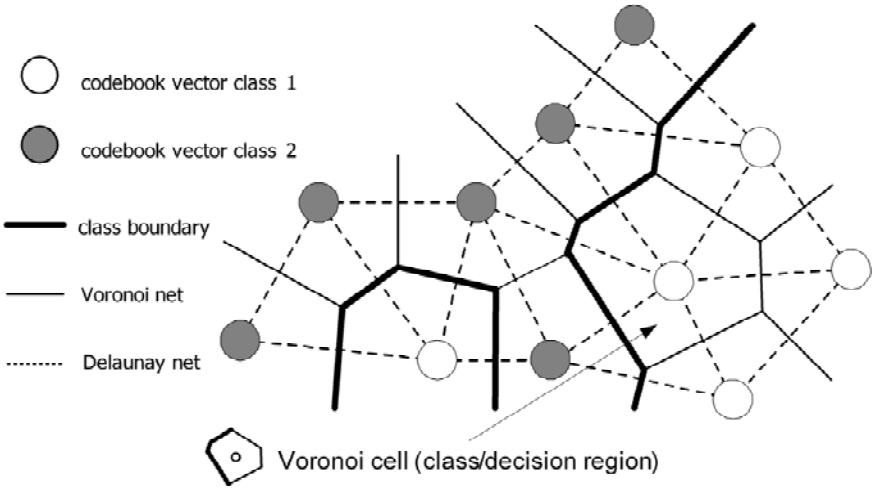

Fig. 4: Tessellation of input space into decision/class regions by codebook vectors represented as neurons positioned in a two-dimensional feature space.

In terms of neural networks a LVQ is a feedforward net with one hidden layer of neurons, fully connected with the input layer. A CV can be seen as a hidden neuron ('Kohonen neuron') or a weight vector of the weights between all input neurons and the regarded Kohonen neuron respectively (see Fig.5).

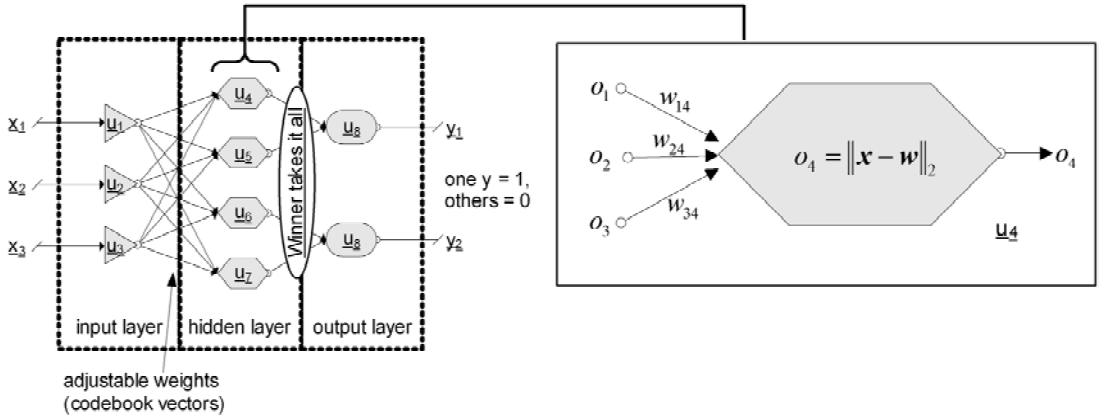

Fig. 5: LVQ architecture: one hidden layer with Kohonen neurons, adjustable weights between input and hidden layer and a winner takes it all mechanism

Learning means modifying the weights in accordance with adapting rules and, therefore, changing the position of a CV in the input space. Since class boundaries are built piecewise-linearly as segments of the mid-planes between CVs of neighbouring classes, the class boundaries are adjusted during the learning process. The tessellation induced by the set of CVs is optimal if all data within one cell indeed belong to the same class. Classification after learning is based on a presented sample's vicinity to the CVs: the classifier assigns the same class label to all samples that fall into the same tessellation – the label of the cell's prototype (the CV nearest to the sample).

The core of the heuristics is based on a distance function – usually the Euclidean distance is used – for comparison between an input vector and the class

representatives. The distance expresses the degree of similarity between presented input vector and CVs. Small distance corresponds with a high degree of similarity and a higher probability for the presented vector to be a member of the class represented by the nearest CV. Therefore, the definition of class boundaries by LVQ is strongly dependent on the distance function, the start positions of CVs, their adjustment rules and the pre-selection of distinctive input features.

The basic LVQ algorithm LVQ1 rewards correct classifications by moving the CV towards a presented input vector, whereas incorrect classifications are punished by moving the CV in opposite direction. The magnitudes of these weight adjustments are controlled by a learning rate which can be lowered over time in order to get finer movements in a later learning phase. Improved versions of LVQ1 are KOHONEN's OLVQ1 with different learning rates for each CV in order to get faster convergence and LVQ2, LVQ2.1 and LVQ3. Since LVQ1 tends to push CVs away from Bayes decision surfaces, it can be expected to get a better approximation of the Bayes rule by pairwise adjustments of two CVs belonging to adjacent classes. Therefore, in LVQ2 adaptation only occurs in regions with cases of misclassification in order to get finer and better class boundaries. LVQ2.1 allows adaptation for correctly classifying CVs, too, and LVQ3 leads to even more weight adjusting operations due to less restrictive adaptation rules. All these algorithms are intended to be applied as extension to previously used (O)LVQ1 (KOHONEN recommends an initial use of OLVQ1 and continuation by LVQ1, LVQ2.1 or LVQ3 with a low initial learning rate). For a comprehensive overview and also details of heuristic learning algorithms of LVQ, readers are referred to standard ANN literature, e.g. [42] or [50]. More detailed information can be found in the above mentioned work of KOHONEN, or for specialized topics, e.g., in [11]. A good overview of statistical and neural approaches to pattern classification is given by [48] or [51]. Besides the above mentioned standard algorithms from KOHONEN, several extensions from various authors are suggested in literature, e.g. LVQ with conscience [34], Learning/Linear Vector Classification (LVC) [45], Dynamic LVQ (DLVQ) [46] or Distinction Sensitive LVQ (DSLVQ) [6]. In [47] LVQ algorithms are discussed and combined with genetic algorithms in order to select and weight useful input features automatically by weighted distances. Newer developments are LVQ4-algorithms with promising performance and results [8].

The accuracy of classification and, therefore, generalisation and the learning speed depend on several factors. Basically the developer of a LVQ has to prepare a learning schedule, a plan which LVQ-algorithm(s) – LVQ1, OLVQ, LVQ2.1 etc. – should be used with which values for the main parameters at different training phases. Also, the number of CVs for each class must be decided in order to reach high classification accuracy and generalisation while avoiding under- or overfitting. Additionally, the rule for stopping the learning process as well as the initialisation method (e.g. random values, values of randomly selected samples) determine the results.

3.4 Support Vector Machines

The original support vector machine (SVM), introduced in 1992 [2, 19, 43], can be characterized as a supervised learning algorithm capable of solving linear and non-linear classification problems. In comparison to neural networks we may describe SVM as a feed-forward neural net with one hidden layer (Fig. 6).

The main building blocks of SVM's are structural risk minimisation, originating from statistical learning theory which was mainly developed by VAPNIK and

CHERVONENKIS [17] , non-linear optimisation and duality and kernel induced features spaces [7, 8], underlining the technique with an exact mathematical framework.

Meanwhile, several extensions to the basic SVM have been introduced, e.g. for multi-class classification as well as regression and clustering problems, making the technique broadly applicable in the data mining area; see for example [47].

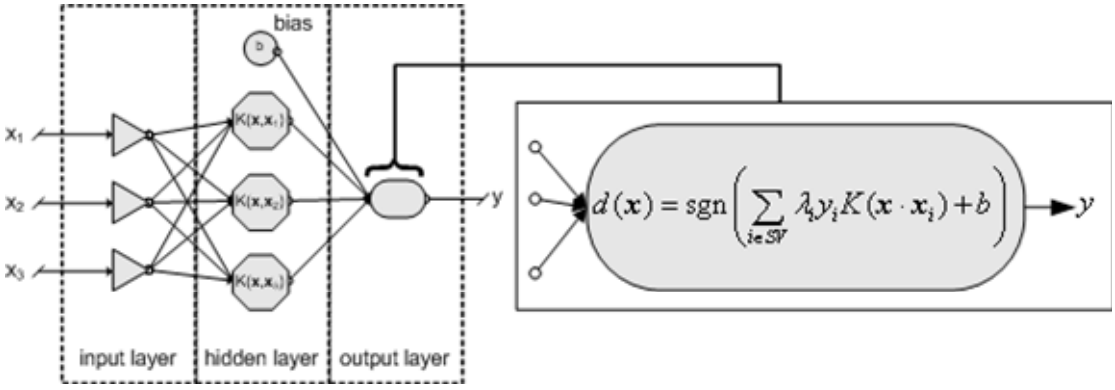

Fig. 6: Architecture of SVM classifier with linear or non-linear kernel function [6].

The main idea of support vector classification is to separate examples with a linear decision surface and maximize the margin between the different classes. This leads to the convex quadratic programming problem (the primal form was omitted for brevity, see for example [8]).

$$\begin{aligned} \text{max.} \quad & W(\lambda) = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \lambda_i \lambda_j y_i y_j \left(\boldsymbol{x}_i \cdot \boldsymbol{x}_j\right) \\ \text{s.t.} \quad & 0 \le \lambda_i \le C \;;\; \sum_{i=1}^{l} \lambda_i y_i = 0 \;\; (i = 1, \ldots l) \end{aligned} \tag{1}$$

The Lagrange multiplier λ_i measures the influence of the i'th learning example on the functional *W*. Examples for which λ_i is positive are called support vectors, as they define the separating hyperplane. *C* is a constant cost parameter, controlling the number of support vectors and enabling the user to control the trade-off between learning error and model complexity, regarded by the margin of the separating hyperplane [41]. As complexity is considered directly during the learning stage, the risk of overfitting the training data is less severe for SVM. The separation rule is given by the indicator function

$$d(\vec{x}) = \mathrm{sgn}\left(\sum_{i \in SV} \lambda_i y_i (\vec{x}_i \cdot \vec{x}) + b\right), \tag{2}$$

using the dot product between the pattern to be classified ($\boldsymbol{x}$), the support vectors and a constant threshold *b*.

For constructing more general non-linear decision functions, SVMs implement the idea to map the examples from input space *X* into a high-dimensional feature space Ψ via an a priori chosen non-linear mapping function. The construction of a separating

hyperplane in the features space leads to a non-linear decision surface in the original space; see Fig. 7. Expensive calculation of dot products in a high-dimensional space can be avoided by introducing a kernel function [5]. Leaving the algorithms almost unchanged, this reduces numerical complexity significantly and allows efficient support vector learning for up to hundreds of thousands examples. The modified decision function is given in Fig. 7.

Thus, the method is very flexible as a variety of learning machines can be constructed simply by using different kernel functions. The conditions a function has to fulfil in order to be applicable as a kernel (Mercer conditions) are described in [44]. Common kernels include polynomials of degree *d* and radial basis function classifiers with smoothing parameter.

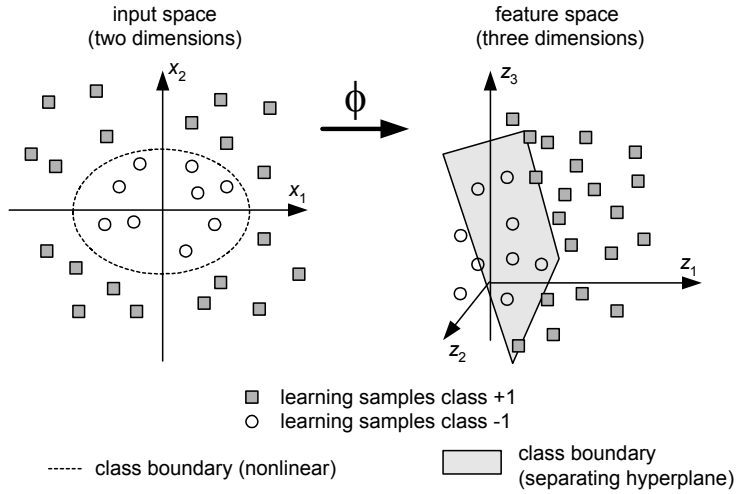

Fig. 7: Non-linear Φ-mapping from two-dimensional input space with non-linear class boundaries into a three-dimensional feature space with linear separation by a hyperplane

Compared to neural networks the SVM method offers a significantly smaller number of parameters. The main modelling freedom consist in the choice of a kernel function and the corresponding kernel parameters, influencing the speed of convergence and the quality of results. Furthermore, the choice of the cost parameter *C* is vital to obtain good classification results, although algorithmic modifications can further simplify this task [16].

4 SIMULATION EXPERIMENT

4.1 Problem description and objectives

The main goal of the empirical simulation experiment is the evaluation of soft computing classification algorithms implemented as SVM, MLP and LVQ in a real world scenario of aCRM. An important objective for a large publishing house is to sell a second subscription to a customer, who has already subscribed one magazine/journal in order to generate additionally profit from a customer relation (‘cross selling’). Therefore, special offers of varying costs are posted to those customers through mailing campaigns in order to exploit their potential cross selling potential, aiming to maximise the response rate in form of the number of new

subscriptions divided by the number of sales letters. By means of response optimisation a presumably optimal group of addressees with the highest anticipated response rate is chosen for the campaign. From the point of aCRM and data mining the problem is to identify a high probability of a second subscription based on attributes of customers with one subscription, e.g. the type of journal already subscribed.

As various classification algorithms are capable of solving problem of this scope, it's unclear which method and which parameterisation is best suited to maximise the response rate. Furthermore, no algorithm can directly operate on raw data and the necessary pre-processing stage offers an even larger variety of degrees of freedom making the overall task even more complicated for the business user. The empirical simulation delivers valuable hints about an appropriate classification technique and its sensitivity with regard to parameterisation and pre-processing issues. Of special interest is the question, if SVMs - quite new to new to the area of data mining and, due to the smaller number of parameters easier to manage - can compete with or even outperform well established techniques like neural networks.

4.2 Experimental design

Following, a description of the selected free modelling parameters for all methods used in the comparative experiments is given. A hold-out method, dividing the data into three separate sets was chosen to control overfitting and allow out-of-sample evaluation.

The available data consisted of 300,000 customer records, which were selected for a previous mailing campaign. The number of subscriptions sold in this campaign was given with 4,019, resulting in a response quote of 1.24%. Handling the extreme dissymmetry in class distributions turned out to be a major challenge of our analysis. Usual approaches to deal with asymmetric class distributions include algorithmic modifications/extensions and advanced sampling strategies. As sampling was inevitable due to the large data set size and because MLP and LVQ do not support asymmetric cost functions natively the latter approach was chosen.

As we are ultimately interested in the minority class of customer who responded in the last mailing, a stratified sampling technique was incorporated to increase the learning machines sensibility for that class. However, stratified sampling introduces another degree of freedom to the experiment, as an appropriate class distribution has to be chosen for the training set (the hold-out set was created by random sampling, ensuring a realistic performance evaluation). A pre-testing stage revealed, that the best classification results where obtained, if positive and negative examples in the training set where evenly distributed. To create data sets of reasonable size, stratified oversampling has been applied to create three disjoint data sets, described in Table 1, which formed the basis for all following experiments.

Table 1: Data set size and structure for the empirical simulation

data set label	data partition		data set usage
training set	20,000	class 1	Used to parameterise all learning algorithms for classification
	20,000	class 0	
validation set	15,000	class 1	Used to supervise the ANNs training process (early stopping).
	15,000	class 0	
generalisation set	1,011	class 1	Hold-out set for out-of-sample evaluation of classifier performance
	73,989	class 0	

Among the vast degrees of freedom in the pre-processing stage, the encoding of categorical attributes, present in almost every aCRM related analysis, and the selection of eligible input variables are most relevant. Therefore, the experimental set-up consists of the combination of three commonly used encoding schemes (N encoding, N-1 encoding and using a single number per categorical attribute) with input and instance selection techniques; see Table 2.

Table 2: Experimental set-up

label	main group	sub group	resulting number of attributes
A.1	single number encoding for categorical attributes	all attributes included	68
A.2		input selection	44
A.3		input selection & outliner filtering	44
B.1	N-1 encoding for categorical attributes	all attributes included	147
B.2		input selection	84
B.3		input selection & outliner filtering	84
C.1	N encoding for categorical attributes	all attributes included	165
C.2		input selection	89
C.3		input selection & outliner filtering	89

Fixing the general experimental framework, several parameterisations for MLP, LVQ and SVM were evaluated and their corresponding performance compared on the generalisation set.

An iterative heuristic approach to determine appropriate architectures (e.g., number of hidden neurons) was selected for ANN. Each network was randomly initialized with 5 to 10 different random seeds to account for alternative starting weights. We selected an early stopping approach, evaluating each network's mean classification rate on a validation set after *r* iterations and stopping the learning process after no increase for *s* iterations (with variations in *r* and *s*). For the MLP, the weighted sum was chosen as the input function and a hyperbolic tangent activation function in all hidden nodes. The output layer used a 1-of-n-code to present two different classes, using a softmax output function with linear activation function.

For LVQ the algorithms LVQ1 and LVQ2.1 were used.

Using a SVM classifier the choice of a network architecture is replaced by selecting an appropriate kernel function [5]. As the application of SVMs to database marketing problems like the one described above is still an ongoing research topic and no kind of prior knowledge was available, we couldn't foresee which kernel would best suit the data. Hence, we selected an iterative approach, evaluating the standard linear,

polynomial and Gaussian kernels with a broad range of common parameter settings as well as symmetric and asymmetric cost functions.

4.3 Visualizing classifier performance

To analyse the influence of pre-processing techniques on classification results, we evaluated classifier performance through the most common metric of classification accuracy, normally calculated as the ratio between correctly classified examples and all examples [40]. A formalisation of this approach leads to the construction of a confusion matrix, a cross-classification of the predicted class against the true class [35].

Table 3: Confusion matrix for binary classification problem with output domain $\{A, B\}$ [5].

		$e(\vec{x})$		
		A	B	$\sum$
y	A	h_{00}	h_{01}	$h_{0.}$
	B	h_{10}	h_{11}	$h_{1.}$
	$\sum$	$h_{.0}$	$h_{.1}$	l

Thus, we can calculate classification accuracy as $\frac{h_{00} + h_{11}}{l}$, with l denoting the size of the evaluation set. However, accuracy based analysis suffer from certain deficits when the underlining class and cost distributions are unbalanced, giving a biased and inappropriate evaluation for practical applications [5, 35].[3] In corporate projects, combining a confusion matrix with case dependant misclassification cost is a valid and straightforward approach, leading to a cost-sensitive measure of classification performance.

However, the technique of receiver operating characteristics (ROC), introduced to the machine learning community by PROVOST and FAWCETT [12], provides a more reliable way to compare classification performance independently of the underlying class distributions. ROC charts are based on the sensitivity (*se*) and specificity (*sp*) of a classifier, which can be derived from the confusion matrix [5]. Evaluating *t* different configurations[4] for each learning machine and calculating the respective pairs (se_t , 1-sp_t), we construct a ROC graph by plotting each point in two dimensional space; see Fig. 8. The optimal region is represented in the upper left corner, with a classifier realizing the furthest upper left point encompasses no errors on the evaluation data set. Thus, ROC analysis provides us with a powerful tool to compare different

[3] The case of response optimisation is a good example, as the exclusion of a customer from a mailing campaign who would otherwise have responded leads to lost sales revenue (h_{10}), whereas the wasted transactions cost for sending direct mail to a prospect who is not interested in (h_{01}), are negligible in comparison.

[4] Here the term "configuration" refers to the combination of degrees of freedom of the learning machines and the pre-processing stage. Thus, a fully parameterized method and a specific way of raw data transformation give one configuration in this sense.

classifiers; drawing each one in ROC space we derive a visual image of their relative performance. Fig. 8 gives an example of two classifiers, where the one represented by the dashed line dominates the other for all possible class and cost distributions.

If the class dependant costs of misclassification are known and deterministic, we may include iso-performance lines in the chart of the ROC-space to choose between different classifiers when there is no clear domination of one method [35]. An extension to ROC analysis, directly representing expected costs, can be found in [36].

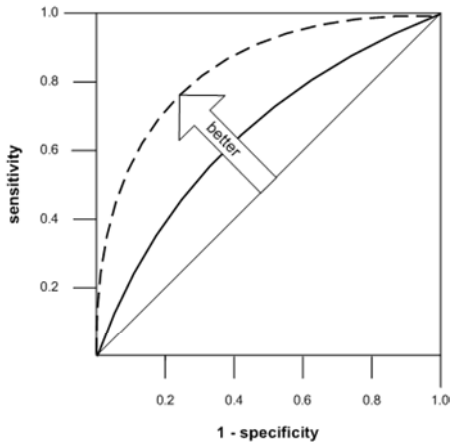

Fig. 8: ROC chart for two binary classifiers showing a situation clear dominance, with the classifier represented by the dashed line outperforming the other classifier for all possible class distributions [27].

4.4 Experimental results

The consolidated main results of the computational experiments are presented in Table 3. Sensitivity and specificity are given for each method applied to the hold-out set after training.

For the application of response optimisation, the performance sensitivity is of predominant importance, as it measures the amount of correctly classified respondents. The sensitivity of SVMs was robustly higher than 50%, with classification rates of 58% regarded as exceptionally good for the application domain. For some MLPs and most LVQs the sensitivity is below 50%, indicating a dominance of the SVM classifier. The classification performance varies from experiment to experiment, proving the considerable influence of pre-processing issues. However, the SVM classifier shows the smallest variance, providing the most robust experimental results.

Table 3: Main results (classification rates on hold-out set [%])

		Group A			Group B			Group C		
		A.1	*A.2*	*A.3*	*B.1*	*B.2*	*B.3*	*C.1*	*C.2*	*C.3*
MLP	sensitivity	49,4	44,8	50,2	51,8	56,0	56,6	18,2	73,0	55,7
	specificity	58,0	61,96	36,5	55,5	55,0	52,9	86,5	38,0	55,4
LVQ	sensitivity	49,8	46,89	53,0	50,0	48,8	39,4	42,5	48,6	34,9
	specificity	55,9	59,1	52,5	55,8	58,9	66,0	72,3	63,7	70,7
SVM	sensitivity	51,6	51,7	50,9	57,47	58,1	54,2	51,0	52,0	55,6
	specificity	60,5	60,35	61,4	56,4	55,6	58,6	56,5	55,9	57,6

Graphing the best SVM, MLP and LVQ classifier for every experiment in ROC-space, this dominance is largely confirmed. Considering the fact, that for any class and cost distribution the optimal classifier has to lie on the boundary of the convex hull, there is only one LVQ result, which could theoretically outperform SVM for a specific class and cost distribution. However, as the corresponding region of the ROC-space has sensitivity below 0.5, it is economically irrelevant. Assuming a cost ratio of $c_0/c_1 = 1/100$ – the profit of a response (and therefore the cost for a customer who was excluded from the campaign but would have responded if included) is hundred times higher than the cost for one mail – the iso-cost line or iso-performance line has a slope of 73,989/101,100 = 0.732 as shown in the ROC chart. The tangential point with the convex hull lies within the SVMs' region and the line is nearly parallel to this SVMs' part of the hull respectively. Lower c_0 or higher c_1 correspond with a lower ratio of the iso-performance line, shifting the tangential point along the convex hull towards the upper right region of the chart (or vice versa).

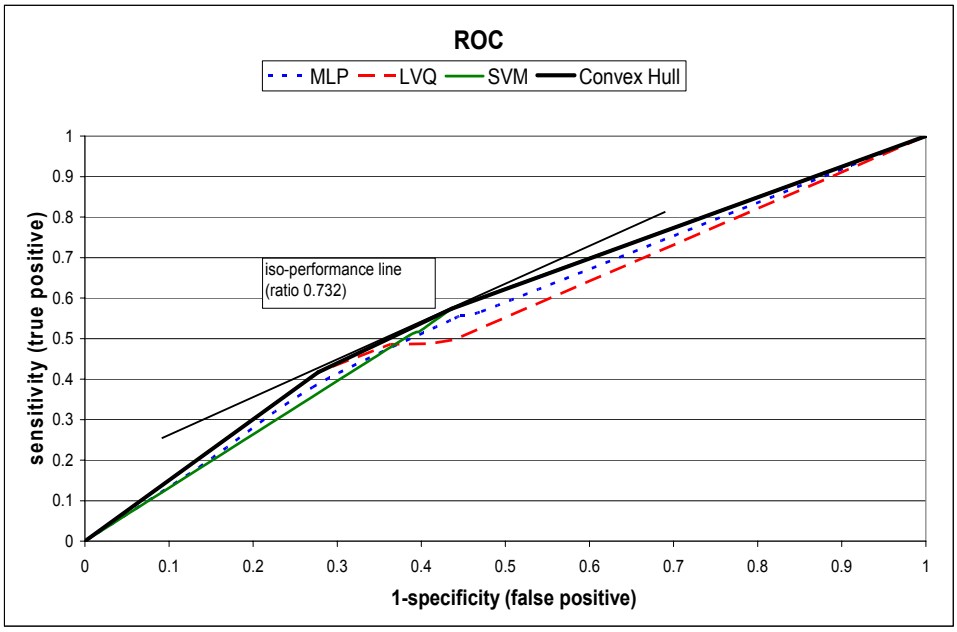

Fig. 4. ROC chart of main results

The radar chart (Fig. 9) shows net profits (profit 100 for each customer minus campaign costs 1 per mail). In almost all experiments (except C.2) the method of SVMs dominate all competing methods. As the quality of LVQs and MLPs varies with the experiments; no clear dominance can be found.

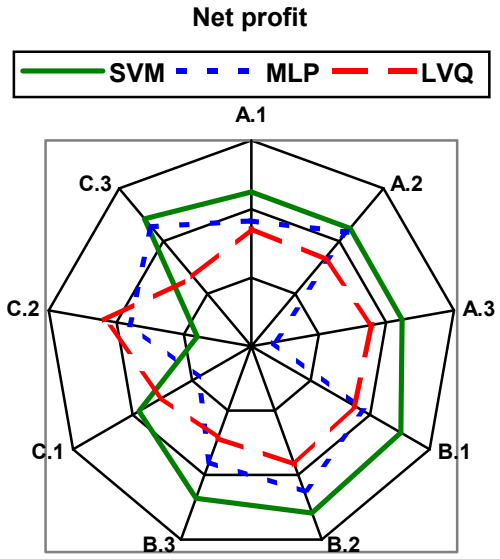

Fig. 9: Radar chart of net profit based on the main results

5 CONCLUSION

Various parameterisation were applied for competing paradigms of ANNs and SVMs. Our numerical results show, that ANN and SVM generally both represent suitable methods for the task of response optimisation, leading to classification accuracy of 58% and more, which can be considered as very good for practical problems.

Preliminary results with various architectures and data pre-processing configurations show severe differences in performance and efficiency between the three evaluated methods. SVM dominate all simulation-results, concurrently delivering robust and superior results, followed by multilayer perceptrons with a specific architecture not found in standard data mining software packages as SPSS Clementine or SAS Enterprise Miner. Consequently, we recommend the integration of SVM algorithms in standard data mining software packages, as the technique is easy to manage and provides competitive results with less parameterisation.

REFERENCES

[1] B. Alex, *Künstliche neuronale Netze in Management-Informationssystemen : Grundlagen und Einsatzmöglichkeiten*. Wiesbaden: Gabler, XXIX, 319 S, 1998.

[2] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support Vector Clustering", *Journal of Machine Learning Research*, vol. 2, pp. 125-137, 2001.

[3] A. Berson, S. Smith, and K. Thearling, *Building Data Mining Applications for CRM*. New York: McGraw Hill, 1999.

[4] C. M. Bishop, *Neural networks for pattern recognition*. Oxford: Clarendon Press, xvii, 482 p., 1995.

[5] T. Bonne and G. Arminger, "Diskriminanzanalyse", in *Handbuch Data Mining im Marketing: Knowledge Discovery in Marketing Databases*, *Business Computing*, H. Hippner, U. Küsters, M. Meyer, and K. Wilde, Eds. Wiesbaden: Vieweg, 2001, pp. 193-240.

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers", *Proc. Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pp. 144-152,
[7] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.
[8] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines : and other kernel-based learning methods*. Cambridge: Cambridge University Press, xiii, 189, 2000.
[9] S. F. Crone, "Training Artificial Neural Networks using Asymmetric Cost Functions", in *Computational Intelligence for the E-Age*, L. Wang, J. C. Rajapakse, K. Fukushima, S.-Y. Lee, and X. Yao, Eds. Singapore: IEEE, 2002, pp. 2374-2380.
[10] S. F. Crone, "Künstliche neuronale Netze zur betrieblichen Entscheidungsunterstützung", *WISU - das Wirtschaftsstudium*, vol. 32, no. 4, pp. 452-458, 2003.
[11] D. DeSieno, "Adding a Conscience to Competitive Learning", *Proc. IEEE International Conference on Neural Networks (ICNN '88)*, vol. I, pp. 117-124, 1988.
[12] C. Drummond and R. C. Holte, "Explicitly Representing Expected Cost: An Alternative to ROC Representation", *Proc. Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 198-207,
[13] L. Fausett, *Fundamentals of Neural Networks : architectures, algorithms, and applications*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[14] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery in Databases : an overview", *AI Magazine*, vol. 17, no. 3, pp. 37-54, 1996.
[15] R. Gray, "Vector quantization", *IEEE ASSP Magazine*, vol. 1, no. 2, pp. 4-29, 1984.
[16] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. Cambridge, Mass. ; London: MIT Press, xxxii, 546 p. : ill. ; 24 cm, 2001.
[17] S. S. Haykin, *Neural networks : a comprehensive foundation*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, xxi, 842, 1999.
[18] J. A. Hertz, A. S. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City: Addison-Wesley, 1991.
[19] C.-W. Hsu and L. Chih-Jen, "A comparison of methods for multi-class support vector machines", *Proc. IEEE Transactions on Neural Networks*, vol. 13, pp. 415-425,
[20] M. Kantardzic, *Data Mining: Concepts, Models, Methods and Algorithms*. New York: John Wiley & Sons, 2003.
[21] N. Kitajima, "A New Method for Initializing Reference Vectors in LVQ", *Proc. IEEE International Conference on Neural Networks (ICNN '95)*, vol. 5, pp. 2775-2779, 1995.
[22] T. Kohonen, *Self-Organizing Maps*, 2 ed. Berlin: Springer, 1997.
[23] K. Koutroumbas, "On the partitioning capabilities of feedforward neural networks with sigmoid nodes", *Neural Computation*, vol. 15, no. 10, pp. 2457-2481, 2003.

[24] U. Küsters, "Data Mining Methoden: Einordnung und Überblick", in *Handbuch Data Mining im Marketing : Knowledge Discovery in Marketing Databases*, H. Hippner, U. Küsters, M. Meyer, and K. Wilde, Eds. Wiesbaden: Vieweg, 2001, pp. 95-130.

[25] J. T. Laaksonen, "A method for analyzing decision regions in Learning Vector Quantization algorithms", *Proc. International Conference on Artificial Neural Networks (ICANN-92)*, vol. 2, pp. 1181-1184, 1992.

[26] S. Lessmann, "Customer Relationship Management", *WISU - das Wirtschaftsstudium*, vol. 32, no. 2, pp. 190-192, 2003.

[27] D. S. Levine, *Introduction to neural and cognitive modeling*, 2nd ed. Mahwah, N.J.: Lawrence Erlbaum Associates Publishers, xix, 491, 2000.

[28] Y. Linde, A. Buzo, and R. Gray, "An Algorithm for Vector Quantizer Design", *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84-95, 1980.

[29] H. Liu and H. Motoda, *Instance selection and construction for data mining*. Boston ; London: Kluwer Academic Publishers, xxv, 416 p. : ill. ; 25 cm, 2001.

[30] M. L. Minsky and S. Papert, *Perceptrons : an introduction to computational geometry*, Expanded ed. Cambridge, Mass.: MIT Press, xv, 292, 1988.

[31] NeuralWare, *Neural Computing - A technology handbook for NeuralWorks Professional II/Plus*. Carnegie, 2001.

[32] NeuralWare, *NeuralWorks Professional II/Plus - Reference Guide*. Carnegie, 2001.

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

[34] M. Pregenzer, D. Flotzinger, and G. Pfurtscheller, "Distinction Sensitive Learning Vector Quantisation - a new noise-insensitive classification method", *Proc. IEEE International Conference on Neural Networks (ICNN '94)*, vol. V, pp. 2890-2894, 1994.

[35] F. Provost and T. Fawcett, "Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions", *Proc. Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pp. 43-48,

[36] R. D. Reed and R. J. Marks, *Neural smithing : supervised learning in feedforward artificial neural networks*. Cambridge, Mass.: The MIT Press, viii, 346 p., 1999.

[37] R. Rojas, *Theorie der neuronalen Netze : eine systematische Einführung*. Berlin: Springer, xviii, 446 p., 1993.

[38] D. E. Rumelhart, J. L. McClelland, and P. R. Group, *Parallel distributed processing : explorations in the microstructure of cognition*. Cambridge, MA: MIT Press, 2 v., 1986.

[39] C. Rygielski, J.-C. Wang, and D. C. Yen, "Data mining techniques for customer relationship management", *Technology in Society*, vol. 24, no. 4, pp. 483-502, 2002.

[40] S. L. Salzberg, "On Comparing Classifiers: Pitfalls to Avoid and a Recommanded Approach", *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 317-328, 1997.

[41] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New Support Vector Algorithms", *Neural Computation*, vol. 12, no. 5, pp. 1207-1245, 2000.

[42] J. Schürmann, *Pattern classification : a unified view of statistical and neural approaches*. New York: Wiley & Sons, 1996.
[43] A. J. Smola and B. Schölkopf, "A Tutorial on Support Vector Regression", Royal Holloway College, University of London, London, 1998.
[44] D. J. Spiegelhalter, C. C. Taylor, and D. Michie, *Machine learning, neural and statistical classification*. New York ; London: Ellis Horwood, xiv, 289p, 1994.
[45] R. Stahlbock, *Evolutionäre Entwicklung künstlicher neuronaler Netze zur Lösung betriebswirtschaftlicher Klassifikationsprobleme*. Berlin: WiKu, 2002.
[46] 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.
[47] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer, 1995.
[48] M. Verleysen, P. Thissen, and J.-D. Legat, "Linear Vector Classification: an improvement on LVQ algorithms to create classes of patterns", *Proc. International Workshop on ANN (IWANN '93)*, pp. 340-345, 1993.
[49] S. M. Weiss and N. Indurkhya, *Predictive data mining : a practical guide*. San Francisco: Morgan Kaufmann Publishers, xii, 228p : ill ; 23cm, pbk, 1998.
[50] T. Y. Young and T. W. Calvert, *Classification, estimation and pattern recognition*. New York: American Elsevier, 1974.
[51] A. Zell, *Simulation Neuronaler Netze*. Munich: R. Oldenbourg Verlag, 624 p., 2000.