Optimizing Hyperparameters of Support Vector Machines by Genetic Algorithms

Stefan Lessmann
Inst. of Business Information Systems
University of Hamburg
D-20146 Hamburg, Germany

Robert Stahlbock
Inst. of Business Information Systems
University of Hamburg
D-20146 Hamburg, Germany

Sven F. Crone
Dep. of Management Science
Lancaster University
Lancaster LA1 4YW, UK

***Abstract*— In this paper, a combination of genetic algorithms and support vector machines (SVMs) is proposed. SVMs are used for solving classification tasks, whereas genetic algorithms are optimization heuristics combining direct and stochastic search within a solution space. Here, the solution space is formed by combinations of different SVM's kernel functions and kernel parameters. We investigate classification performance of evolutionary constructed SVMs in a complex real-world scenario of direct marketing.**

Keywords: artificial support vector machine, classification, genetic algorithm, data mining, decision support

I. Introduction

In this paper we investigate the application of evolutionary constructed SVMs on an economic real-world classification problem. The genetic component provides improved selection and parametrization of SVM's essential kernel function in order to achieve higher classification accuracy.

The organization of this paper is as follows. In Section II, the task of classification and the classification method SVM in its standard form are briefly presented. In Section III the main principles of genetic algorithms are proposed. Both ideas are combined in Section IV. The computational experiment for solving the real-world classification task of response optimization in direct marketing is specified in Section V. Results and analyses are presented in Section VI. Section VII concludes the paper with a summarization.

II. Classification and Support Vector Machines

Data driven methods from computational intelligence share the common approach of learning machines in classification for data mining [1]. Let all relevant and measurable attributes of an object, e.g. a customer, be combined as numerical values in $\vec{x}$. Let the the input space with n objects be denoted in the set $\mathbb{X} = \{\vec{x}_1, \ldots, \vec{x}_n\}$. Each object belongs to a discrete class $y \in \{-1; 1\}$ and we will refer to a pair $(\vec{x}, y)$ as an example of our classification problem. We presume that it is impossible to model the relationship between attribute vector $\vec{x}$ and class membership y directly, either because it is unknown, too complex or corrupted by noise. Presuming that a sufficient large set of examples is available, a machine for supervised learning of the mapping $\vec{x} \rightarrow y$ can be incorporated.

The objective of a classification process is to modify free parameters in order to find a specific learning machine for modelling a generalizable causal relationship of the problem structure from the learning data to predict unseen examples based on their attribute values x_j. For most questions of parametrization only rules of thumb are known. For a comprehensive discussion readers are referred to e.g. [1–4].

A SVM in its basic form can be characterized as a supervised learning machine for solving linear and non-linear classification tasks. In a modified form a SVM can solve regression tasks as well. It was introduced by Vapnik based on fundamentals of statistical learning theory and risk minimization [3–7]. Main building blocks of SVMs are structural risk minimization, non-linear optimization and duality as well as kernel induced feature spaces, underlining the technique with an exact mathematical framework.

The idea of support vector classification is to separate examples with a linear decision surface, maximizing the margin of separation between two different classes. This leads to the following convex quadratic programming problem – the primal form was omitted for brevity, see e.g. [3].

$$\begin{aligned} max\ W(\lambda) &= \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j (\vec{x}_i \cdot \vec{x}_j) \\ s.t. \quad & 0 \leq \lambda_i \leq C \quad \forall i = 1, \ldots, n \\ & \sum_{i=1}^{n} \lambda_i y_i = 0 \end{aligned} \tag{1}$$

Examples with positive Lagrangian multiplier λ_i are called support vectors (SV). They define the separating hyperplane. C is a constant regularization parameter, enabling the user to control the trade-off between learning error and model complexity, regarded by the margin of the separating hyperplane. The decision function for classification can be evaluated in terms of dot products between the pattern to be classified and the support

vectors with b being a threshold:

$$f(\vec{x}) = \operatorname{sgn}\left(\sum_{i \in \mathrm{SV}} \lambda_i y_i (\vec{x} \cdot \vec{x}_i) + b\right). \quad (2)$$

For constructing more general non-linear decision surfaces, SVMs implement the idea of mapping the input vectors into a high-dimensional feature space Ψ via an a priori chosen non-linear mapping function $\Phi : X \to \Psi$. The construction of a separating hyperplane in feature space leads to a non-linear decision boundary in input space. Expensive calculation of dot products $\Phi(\vec{x}) \cdot \Phi(\vec{x}_i)$ in a high-dimensional space can be avoided by introducing a kernel function $k(\vec{x}, \vec{x}_i) = \Phi(\vec{x}) \cdot \Phi(\vec{x}_i)$. An illustration of a Φ-transformation from two-dimensional input space with non-linear class boundary into three-dimensional feature space enabling linear separation with a hyperplane is given in Fig. 1.

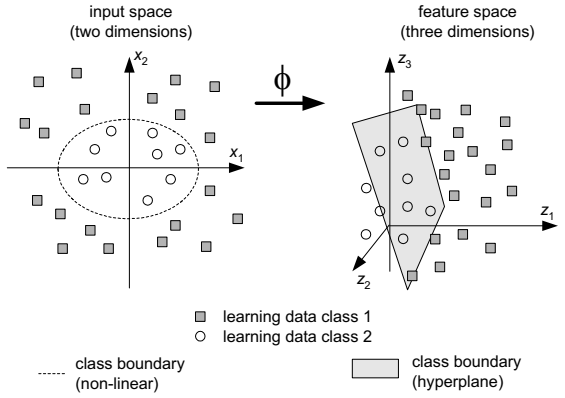

Fig. 1: Example of Φ-transformation

The kernel representation allows computing the dot product's value in feature space without explicit calculation of the mapping Φ. Leaving the algorithms almost unchanged, this reduces numerical complexity significantly and allows efficient support vector learning for up to hundreds of thousand examples. Degrees of freedom are significantly smaller for SVM, compared to typical neural network paradigms like, e.g., multilayer perceptrons. Furthermore, neural network architectures can be modelled by SVMs with sigmoid or radial kernel functions. Using SVM, the main freedom is the choice of a kernel function with its 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 for learning data as well as for hold-out data.

III. Genetic Algorithms

Genetic algorithms (GA), a subgroup of evolutionary algorithms (EA), are meta-heuristics imitating the long-term optimization process of the biological evolution for solving mathematical optimization problems. They are based upon Darwin's idea of '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. Within GA, reproduction and variation is achieved not only by mutation of genes but also by the crossover operator. The latter combines characteristics of two solutions in order to get two 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. Moreover, the fitness function has great impact on performance (see e.g. [8–10] for an overview and more detailed criteria for efficient promising GA approaches, e.g. completeness, compactness and short schemata).

IV. Support Vector Machines in Combination with Genetic Algorithms

In conjunction with SVM, EA can be used for solving problems like, e.g., construction of an appropriate kernel function, optimization of the SVM's parameters and/or for determination of the most reasonable input data. Combinations of EA and SVM for solving classification tasks can be found e. g. in [11–15]. We develop a new GA-SVM, extending the approach of Ohn et al. [11–13].

In our experiment the kernel function and its parameters are subject of evolution. We construct the kernel as combination of five possible kernels with an additive or a multiplicative operator. The five basic kernel functions are

- polynomial
  $k_{\mathrm{poly}}(\vec{x}, \vec{x}_i) = (a\vec{x} \cdot \vec{x}_i + b)^c$
- radial
  $k_{\mathrm{rad}}(\vec{x}, \vec{x}_i) = \exp(-a \|\vec{x} - \vec{x}_i\|^2)$
- sigmoid
  $k_{\mathrm{sig}}(\vec{x}, \vec{x}_i) = \tanh(a(\vec{x} \cdot \vec{x}_i) + b)$
- inverse multi-quadratic
  $k_{\mathrm{imq}}(\vec{x}, \vec{x}_i) = 1/\sqrt{\|\vec{x} - \vec{x}_i\|^2 + b^2}$
- anova
  $k_{\mathrm{anova}}(\vec{x}, \vec{x}_i) = \left(\sum_j \exp(-a(\vec{x}_j - \vec{x}_{i_j})^2)\right)^c$

The sum as well as the product of two kernels is a new valid kernel fulfilling the necessary Mercer conditions [3]. Therefore, our constructed kernel function has the following structure with $\otimes \in \{+, \cdot\}$:

$$k^1_{\mathrm{poly}} \otimes k^\alpha_{\mathrm{rad}} \otimes k^\beta_{\mathrm{sig}} \otimes k^\gamma_{\mathrm{imq}} \otimes k^1_{\mathrm{anova}}.$$

We have the following encoding of genotype, chromosomes and genes, respectively:

- chromosome for exponents of each kernel function (5 integer genes)
  Each gene can encode an integer in the interval of $[0,\ldots,7]$ except genes for polynomial and anova kernel having an exponent naturally. Those coefficients are fixed to 1.
- chromosome for operator $\otimes$ aggregating two kernels by addition or multiplication (4 binary genes)
- chromosome for up to three parameters a,b,c for each kernel (3 decimal genes), resulting in a total of 15 genes
- chromosome for regularization parameter C (1 decimal gene)

The complete structure of the genotype with its chromosomes and genes is illustrated in Fig. 2.

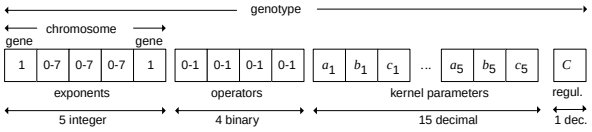

Fig. 2: Chromosomes with genes

The GA-based development of SVMs is an iterative process. A SVM is constructed by transferring the genotype's genetic code into a phenotype, i.e. a SVM with a well defined kernel function and valid parameters. After learning and (cross-)validation, a SVM is evaluated by a fitness function. Genetic operations use this quality information for building a new population of SVMs, which are trained and evaluated again. Thus, the whole learning process can be seen as subdivided into a microscopic cycle for learning of a SVM and a macroscopic evolutionary one. This process (Fig. 3) is principally quite similar to the process of evolutionary constructing artificial neural nets (see e.g. [16, 17]).

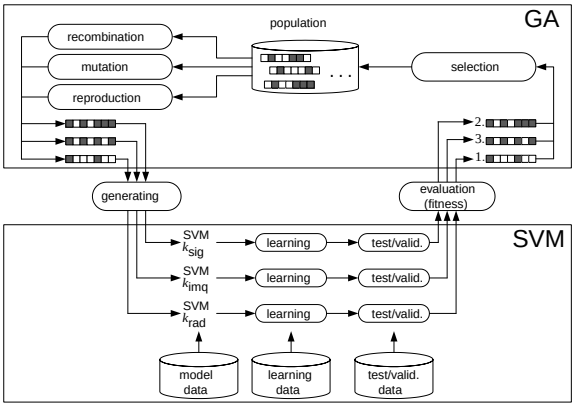

Fig. 3: Evolution of SVMs by GA

The superordinated evolutionary learning process is guided by the fitness function representing the user's objective. Thus, the formulation of an appropriate fitness function highly depends on the specific problem to be solved. Since our dataset incorporates severely imbalanced class distributions we use the geometric mean G of the accuracy on class -1 and class 1, i.e. the classification rate on each class for evaluation of classification performance of an individual SVM. [18, 19]:

$$G = \sqrt{CR_{(-1)} \cdot CR_{(1)}} \tag{3}$$

with $CR_{(y)} = n_y^+ / m_y$

n_y^+ denoting the number of patterns of class y that are classified as member of class y (correct)

m_y denoting the number of all patterns belonging to class y

Results obtained on the training and validation set are used to guide the search of the GA. In general, it is desirable that a classifier separates the training set with few errors whereas an additional validation set is required to prevent overfitting. To incorporate these ideas into a single metric we combine these G-values on learning and validation datasets by calculating their arithmetic mean within our fitness function for GA:

$$F = \frac{G^{\text{learn}} + G^{\text{validation}}}{2}. \tag{4}$$

Whereas the fitness function is used for selection of solutions for reproduction, the reproduction itself is conducted by means of mutation and crossover. The selection is implemented as tournament selection with a tournament size 2. Furthermore, an elitist mechanism is applied in order to ensure that the best SVM is member of the next generation. The crossover operator is implemented as uniform crossover, i.e. all genes between two random points within a chromosome are interchanged between two genotypes representing parents for the resulting two new genotypes. [20, 21]

Crossover is potentially applied to chromosomes for kernel aggregation and kernel exponent, whereas mutation can be applied to all chromosomes. The actual application of a genetic operation depends on user-defined rates. A high rate for crossing over and low rate for mutation are recommended. We set the crossover rate to 0.7 and the mutation rate for one gene to 0.3. Mutation is implemented as a stepwise increment or decrement with a specific stepsize resulting in a new value within minimum and maximum limits. Binary genes are mutated by flipping 0 to 1 and vice versa.

V. Computational Experiment – Response Optimization

The simulation experiments aim at empirical evaluation of the improvements of SVMs by means of GA. Without doubt, SVMs provide excellent results for classification tasks in different fields of application. Furthermore, GAs are well established for optimization tasks. What is the impact of a combination of these two algorithms? From the point of theory, results of GA-SVM should outperform manually constructed SVMs due to evolution-based intelligent search heuristics. From the point of practice, the high computational power being necessary for computation of enough phenotypes in order to get a meaningful evolutionary process might be the main obstacle for applications in time-critical fields.

The available experimental data consist of 300,000 customer records of a large German publishing house, which were selected for a past mailing campaign. The campaign aimed at cross-selling an additional magazine subscription to existing customers with at least one periodical subscription. The number of subscriptions sold in that campaign is given with 4,019, resulting in a response quote of 1.34 %. This quote is deemed representative for the application domain. The data set of 28 variables contains 9 numerical, continues attributes and 19 categorical attributes describing demographic and transaction oriented customer properties. An additional binary target variable indicates if the respective customer subscribed to a magazine after receiving the mailing (class 1) or not (class -1). In order to select profitable customers for an upcoming campaign, i.e. customers with high probability of responding to a solicitation, it is common practice to use supervised learning algorithms to predict the class membership of each customer.

All experiments are evaluated applying a hold-out method, using three disjoint datasets in order to control over-fitting and allow out-of-sample evaluation. While training data is used for learning, i.e. parameterizing the methods to adjust class boundaries, a second set is used for validating the classifiers' performances to guide the learning process and for model selection. The trained and selected classifiers are finally tested on an unknown hold-out set to evaluate their generalization ability similar to estimate their performance in a real world scenario on unknown data.

The three data sets are constructed as follows. At first, a set of 65,000 records was randomly chosen out of the total 300,000 records, building an isolated hold-out set for generalization test. Statistically, this set should include approximately 1.34 % (=871) responders (class 1). In fact the set is partitioned into 912 members of class 1 and 64,088 members of class -1. The size of this set accounts for a reasonable high number of responders' records being in the learning and test sets used for model building as well as for keeping the record number not too high in order to save computing time. The second step concentrates on full usage of the remaining 3,107 responders. Two-thirds (2,072) are randomly chosen and assigned to the learning set. The rest (1,035) forms a part of the validation set. Since the ratio between class 1 customers and class -1 customers in the data set is heavily skewed we made use of undersampling for equating the number of randomly chosen records of inactive customers with the number of responders in each set. The idea of undersampling is to increase the ratio of the minority class in the learning set to increase the classifier's sensitivity for this class [22]. This approach creates an evenly balanced learning set of 4,144 records and a validation set of 2,070 records. Table I shows the resulting sizes and structures of the data set.

Table I: Data set structure and size for simulation experiments

data set label	number of records	class	data set usage
learning	2,072	1	set parameters of all learning algorithms
	2,072	-1	
validation	1,035	1	supervise training process
	1,035	-1	
hold-out	912	1	out-of-sample evaluation of classifier performance (generalization)
	64,088	-1	

Using a single number encoding approach, all categories are transformed to numerical values in an initial pre-processing step. Following, all attributes are scaled into the interval [-1,1] to account for different measurement levels and number ranges. [23–25]

VI. Experimental Results

In order to deliver good results GA usually requires a large population size that ensures sufficient variability within the elements in the gene pool. For GA-SVM we select a population size of 50 and monitor the progress in classification quality by means of the *G*-metric for 16 generations. Thus, $16 \cdot 50 = 800$ individual SVMs with genetic kernel are constructed on the training set, assessed on the validation set and finally evaluated on the test set. Detailed results on the generation level are given in Table II on the next page. The first column of Table II gives the number of the respective GA generation. Following, the mean runtime per SVM as well as mean, maximum, minimum and standard deviation of the *G*-metric are reported. These were calculated on the basis of the 50 individual GA-SVM classifiers within

Table II: Results of GA-SVM on generation level

Generation	Mean runtime per SVM [min]	*G*-metric on training/validation set mean	max	min	std.dev.	*G*-metric on generalization set mean	max	min	std.dev.	Similarity
0	47.64	0.6820	0.89	0.08	0.2204	0.3539	0.53	0.09	0.1256	1.7001
1	46.57	0.7467	0.89	0.06	0.1942	0.3984	0.53	0.05	0.1034	2.2817
2	46.51	0.8229	0.89	0.49	0.1102	0.4373	0.51	0.34	0.0356	1.8231
3	47.67	0.8583	0.89	0.52	0.0794	0.4337	0.51	0.22	0.0381	1.8398
4	49.58	0.8749	0.89	0.70	0.0305	0.4359	0.48	0.35	0.0152	1.8166
5	50.75	0.8391	0.89	0.32	0.1298	0.4251	0.53	0.14	0.0634	1.9767
6	48.23	0.8736	0.89	0.67	0.0489	0.4298	0.50	0.12	0.0541	1.5291
7	46.79	0.8441	0.89	0.51	0.0988	0.4409	0.51	0.22	0.0427	2.1306
8	46.57	0.8588	0.89	0.55	0.0749	0.4436	0.50	0.40	0.0200	2.3792
9	47.14	0.8616	0.89	0.51	0.0811	0.4449	0.51	0.42	0.0190	2.1037
10	47.95	0.8539	0.89	0.51	0.0954	0.4400	0.53	0.30	0.0316	2.3666
11	47.31	0.8626	0.89	0.58	0.0674	0.4372	0.52	0.22	0.0392	2.2904
12	47.85	0.8670	0.89	0.62	0.0483	0.4363	0.52	0.22	0.0368	1.6356
13	46.84	0.8451	0.89	0.52	0.1030	0.4416	0.52	0.28	0.0334	1.7407
14	49.61	0.8518	0.89	0.52	0.0910	0.4443	0.50	0.41	0.0185	2.2396
15	47.33	0.8457	0.89	0.51	0.0985	0.4350	0.50	0.17	0.0501	2.5081

each population. While the results in the first four columns reflect the combined performance on training and validation set (mean *G*-metric (3)) the subsequently assessments were obtained on the generalization set.

Noteworthy, the maximum *G*-metric on learning/validation is identical over all generations. This originates from our elitist selection, because a high quality kernel was found by chance in the first generation. The evolution of the mean *G*-values on training/validation and generalization set is plotted in Fig. 4. The chart supports the impression that good results are obtained early in the process. Consequently, the improvements in performance are highest in the first three generations and then reach a saturation level without further advancements.

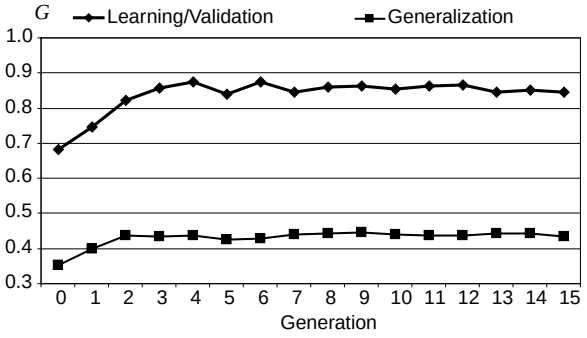

Fig. 4: *G*-values for GA-SVM

To verify if this development is due to increased similarity among the genotypes within the population we calculate a similarity measure (last column in Table II) which is based on the standard deviation between the Euclidian norms of the chromosome vectors within the respective generation. Similarity values are scaled into interval $[0,4]$ for absolute similarity (0) to absolute dissimilarity (4), so that we observe a moderate similarity in each generation without a clear trend. Hence, further research is needed to clarify whether GA-SVM usually requires only a few generations or whether this result is valid for the considered data set only.

To evaluate the effectiveness of GA-SVM as a technique for model selection we compared our results to a standard SVM classifier with radial kernel function. Conducting a simple grid search [26] in the range $\log(C) = \{-2,1,0,1,2\}$ and $\log(\gamma) = \{-2,1,0,1,2\}$ we calculated 25 solutions for the standard SVM on the same data sets; results for generalization set are given in Table III. Results on learning/validation set are omitted for brevity but show a similar trend.

Standard SVMs' mean generalization set *G*-value is always lower than those of GA-SVM. A one-tailed pooled variance T-Test revealed that this inferiority is statistically significant at the 0.05 level in nine out of fifteen cases. A previously conducted F-test confirmed the assumption of identical variances. Another important criterion beside test set effectiveness is the stability of the resulting classifier. In a real world application the decision maker selects the classifier with highest validation set performance to predict the class membership of new examples. Consequently, a high correlation between validation and test set performance is required to ensure that the right classifier will be selected in this scenario. Since the Pearson correlation coefficient between learning/validation set *G*-value and generalization set *G*-value for standard SVM is again always lower than those of GA-SVM we can conclude that GA-SVM not only delivers more accurate results but also more stable ones.

Table III: Results of GA-SVM – grid search – generalization set

Generation	Generalization set results Mean G-value GA-SVM	SVM	Significance (0.05 level)	Correlation learning/validation vs. generalization GA-SVM	SVM
0	0.3539		0.3454	0,8245	
1	0.3984		*0.0284*	0.7154	
2	0.4373		*0.0332*	0.8818	
3	0.4337		*0.0183*	0.8939	
4	0.4359		*0.0351*	0.9849	
5	0.4251		0.1670	0.9045	
6	0.4298		*0.0103*	0.9304	
7	0.4409	0.3274	*0.0124*	0.8332	0.6524
8	0.4436		0.1318	0.9535	
9	0.4449		*0.0214*	0.9187	
10	0.4400		0.0612	0.8979	
11	0.4372		0.1074	0.9452	
12	0.4363		*0.0156*	0.9442	
13	0.4416		0.0911	0.9216	
14	0.4443		0.2075	0.7814	
15	0.4350		*0.0185*	0.8709	

italic: significant at 0.05 level

VII. Conclusion

In this paper we extended state-of-the-art approaches for SVMs with evolutionary optimized kernel function including additional kernel functions and SVMs regularization parameter in the genetic kernel. Solving more than 800 evolutionary constructed SVMs we find that our GA-SVM provides indeed more accurate and stable results than standard SVM with radial kernel. While this finding confirms previous results with GA-SVM this is to our best knowledge the first attempt to apply GA-SVM to a real world scenario of corporate decision making. However, the improvements of the method come at a high computational cost. While the standard SVM needs less than a minute to construct a solution, the average run-time of GA-SVM lies between 45 and 50 minutes (P4, 3 GHz, 1 GB RAM). The computationally expensive evaluation of the genetic kernel questions the applicability of GA-SVM in large scale application. To some extend, the parallelization of GA-SVM could help to mitigate this shortcoming but would increase the complexity of the overall infrastructure. The possible potential of new techniques for efficient calculation and caching of genetic kernels has to be investigated in further research.

References

[1] S. S. Haykin, *Neural networks : a comprehensive foundation*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press, 1995.

[3] N. Cristianini and J. Shawe-Taylor, Eds., *An Introduction to Support Vector Machines : and other kernel-based learning methods*. Cambridge, UK: Cambridge University Press, 2000.

[4] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge: Cambridge University Press, 2004.

[5] V. Vapnik, "Learning and Generalization: Theoretical Bounds," in *The handbook of brain theory and neural networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995, pp. 516–522.

[6] ——, *Statistical Learning Theory*. New York: Wiley, 1998.

[7] ——, *The Nature of Statistical Learning Theory*, 2nd ed. New York: Springer, 1999.

[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[9] 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.

[10] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 2nd ed. Berlin: Springer, 1994.

[11] H.-N. Nguyen, S.-Y. Ohn, and W.-J. Choi, "Combined kernel function for support vector machine and learning method based on evolutionary algorithm," in *Neural Information Processing: 11th International Conference - Proc. of ICONIP 2004, Calcutta, India, Nov 22-25, 2004*. Heidelberg: Springer, 2004, pp. 1273–1278, dOI 10.1007/b103766.

[12] S.-Y. Ohn, H.-N. Nguyen, D. S. Kim, and J. S. Park, "Determining optimal decision model for support vector machine by genetic algorithm," in *Computational and Information Science: First International Symposium - Proc. of CIS 2004, Shanghai, China, Dec 16-18, 2004*. Heidelberg: Springer, 2004, pp. 592–597, dOI 10.1007/b104566.

[13] S.-Y. Ohn, H.-N. Nguyen, and S.-D. Chi, "Evolutionary parameter estimation algorithm for combined kernel function in support vector machine," in *Content Computing: Advanced Workshop on Content Computing – Proc. of AWCC 2004, ZhenJiang, JiangSu, China, Nov 15-17, 2004*. Heidelberg: Springer, 2004, pp. 592–597, dOI 10.1007/b103383.

[14] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple svm parameters," *Neurocomputing (in press)*, 2005.

[15] D. Eads, D. Hill, S. Davis, S. Perkins, J. Ma, R. Porter, and J. Theiler, "Genetic algorithms and support vector machines for time series classification," in *Proc. of Fifth Conf. on the Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation. Symposium on Optical Science and Technology of the 2002 SPIE Annual Meeting, Seattle, WA, July 2002.*, 2002, pp. 74–85, lA-UR-02-6212.

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

[17] ——, "An evolutionary neural classification approach to evaluate retail stores and support decisions on their location, in-store

design and assortment," in *Proc. of the Int. Conf. on Artificial Intelligence (IC-AI'04), Las Vegas, Nevada, USA, June 21–24*, H. Arabnia, Ed., vol. I. CSREA Press, 2004, pp. 228–234.

[18] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine Learning*, vol. 30 (2–3), pp. 195–215, 1998.

[19] S. Lessmann, "Solving imbalanced classification problems with support vector machines," in *Proc. of the Int. Conf. on Artificial Intelligence (IC-AI'04), Las Vegas, Nevada, USA, June 21–24*, H. Arabnia, Ed., vol. I. CSREA Press, 2004, pp. 214–220.

[20] V. Nissen, *Evolutionäre Algorithmen : Darstellung, Beispiele, betriebswirtschaftliche Anwendungsmöglichkeiten*. Wiesbaden: Dt. Univ.-Verl., 1994.

[21] ——, *Einführung in Evolutionäre Algorithmen : Optimierung nach dem Vorbild der Evolution*. Braunschweig: Vieweg, 1997.

[22] N. Japkowicz and S. Stephen, "The Class Imbalance Problem: A Systematic Study," *Intelligent Data Analysis*, vol. 6 (5), pp. 429–450, 2002.

[23] D. Pyle, *Data preparation for data mining*. San Francisco: Morgan Kaufmann, 1999.

[24] M. J. Berry and G. Linoff, *Data mining techniques : for marketing, sales and customer relationship management*, 2nd ed. New York: Wiley, 2004.

[25] S. F. Crone, S. Lessmann, and R. Stahlbock, "Empirical comparison and evaluation of classifier performance for data mining in customer relationship management," in *Proc. of IEEE 2004 International Joint Conference on Neural Networks (IJCNN 2004), Budapest, Hungary, July 25–29*, D. Wunsch, Ed., vol. 1. IEEE, 2004, pp. 443–448.

[26] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviours of support vector machines with gaussian kernel," *Neural Computation*, vol. 15, pp. 1667–1689, 2003.