Lecture Notes in Computer Science Vol. 7267, pp. 412-420, 2012.

Locally Optimized Kernels

Tomasz Maszczyk and Włodzisław Duch

Department of Informatics, Nicolaus Copernicus University
Grudziądzka 5, 87-100 Toruń, Poland
{tmaszczyk,wduch}@is.umk.pl
http://www.is.umk.pl

Abstract. Support Vector Machines (SVM's) with various kernels have become very successful in pattern classification and regression. However, single kernels do not lead to optimal data models. Replacing the input space by a kernel-based feature space in which the linear discrimination problem with margin maximization is solved is a general method that allows for mixing various kernels and adding new types of features. We show here how to generate locally optimized kernels that facilitate multi-resolution and can handle complex data distributions using simpler models than the standard data formulation may provide.

1 Introduction

For more than a decade now kernel Support Vector Machines (SVM's) have become extremely successful approaches to pattern classification and regression problems. Excellent results have been reported in applying SVM's in multiple domains thanks to the ingenious use of various kernels, providing an equivalent of specific similarity measures in the kernel space [1]. However, the type of solution offered by a given data model obtained by SVM with a specific kernel may not be the most appropriate for particular data. Each data model defines a hypotheses space, that is a set of functions that this model may easily learn. Linear methods work best when decision borders are flat, but they are obviously not suitable for spherical distributions of data. For some problems (for example, high-dimensional parity and similar functions), neither linear nor radial decision borders are sufficient [2].

Kernel methods implicitly provide new, useful features $z_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$ constructed around support vectors $\boldsymbol{x}_i$, a subset of input vectors relevant to the training objective. Prediction is supported by new features, most often distance functions from selected training vectors, weighted by a Gaussian function, making the decision borders flat in the kernel space. Multiple kernels may be used to construct new features, as shown in our Support Feature Machine algorithm [3]. In the second section standard approach to the SVM is described and linked to evaluation of similarity to support vectors in the space enhanced by $z_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$ kernel features. Linear models defined in the enhanced space are equivalent to kernel-based SVMs. In particular, one can use linear SVM to find discriminant in the enhanced space, preserving the wide margins. For special problems other linear discriminant techniques may be more appropriate [4].

Although most research in the SVM community has focused on the underlying learning algorithms the study of kernels has also gained importance recently. Standard

kernels such as linear, Gaussian, or polynomial do not take full advantage of the nuances of specific data sets. This has motivated plenty of research into the use of alternative kernels. Various kernels may be used to create enhanced feature space. Here an approach that combines Gaussian kernels with selection of pure clusters, adopted from our Almost Random Projection Machines [5] algorithm, is investigated. In section 4 Locally Optimized Kernels are tested in a number of benchmark calculations. Brief discussion of further research directions concludes this paper.

2 Kernels and Support Vector Machines

2.1 Standard SVM formulation

Since the seminal paper of Boser, Guyon and Vapnik in 1992 [6] Support Vector Machines quickly became the most popular method of classification and regression, finding numerous other applications [7–9]. In case of binary classification problems SVM algorithm minimizes average errors (or risk) over the set of data pairs $\langle x_i, y_i \rangle$. Depending on the choice of kernels and optimization of their parameters SVM can produce flexible nonlinear data models that, thanks to the optimization of classification margin, offer good generalization. This means that the minimum distance between the training vectors $\boldsymbol{x}_i$ and the hyperplane $\boldsymbol{w}$ should be maximized:

$$\max_{\boldsymbol{w},b} \min \|\boldsymbol{x} - \boldsymbol{x}_i\| \ : \ \boldsymbol{w} \cdot \boldsymbol{x} + b = 0, \ i = 1, \ldots, m \tag{1}$$

The $\boldsymbol{w}$ and b can be rescaled in such a way that the point closest to the hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$, lies on one of the parallel hyperplanes defining the margin $\boldsymbol{w} \cdot \boldsymbol{x} + b = \pm 1$. This leads to the requirement that

$$\forall_{\boldsymbol{x}_i} \ y_i[\boldsymbol{w} \cdot \boldsymbol{x}_i + b] \geq 1 \tag{2}$$

The width of the margin is equal to $2/\|w\|$. Therefore maximization of margins is equivalent to minimization:

$$\min_{\boldsymbol{w},b} \ \tau(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{3}$$

with constraints that guarantee correct classification:

$$y_i[\boldsymbol{w} \cdot \boldsymbol{x}_i + b] \geq 1 \quad i = 1, \ldots, m \tag{4}$$

Constraint optimization problems are solved by defining the Lagrangian:

$$L(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i[\boldsymbol{x}_i \cdot \boldsymbol{w} + b] - 1) \tag{5}$$

where $\alpha_i > 0$ are Lagrange multipliers. Its minimization over b and $\boldsymbol{w}$ leads to two conditions:

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \quad \boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \tag{6}$$

The vector $\boldsymbol{w}$ that defines the hyperplane is expressed as a combination of the training vectors, each component $\boldsymbol{w}[j]$ is a combination of j feature values for all vectors $\boldsymbol{x}_i[j]$. According to the Karush-Kuhn-Thucker conditions:

$$\alpha_i(y_i[\boldsymbol{x}_i \cdot \boldsymbol{w} + b] - 1) = 0, \quad i = 1, \ldots, m \tag{7}$$

For $\alpha_i \neq 0$ vectors must lie on one of the margin hyperplanes $y_i[\boldsymbol{x}_i \cdot \boldsymbol{w} + b] = 1$; these vectors "support" the hyperplane $\boldsymbol{w}$ that defines the solution of the optimization problem. Although the minimization may be performed in the primal form [10] the quadratic optimization problem is frequently redefined in a bit simpler dual form:

$$\max_{\alpha} \boldsymbol{w}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \boldsymbol{x}_j \tag{8}$$

with constraints:

$$\alpha_i \geq 0 \;\; i = 1, \ldots, m \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{9}$$

The discriminant function takes the form:

$$g(x) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x} \cdot \boldsymbol{x}_i + b \right) \tag{10}$$

Now it is easy to replace dot product $\boldsymbol{x} \cdot \boldsymbol{x}_i$ by a kernel function $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}')$ where $\phi(\boldsymbol{x})$ represents an implicit transformation of the original vectors to a new kernel space. For any $\phi(\boldsymbol{x})$ vector the part orthogonal to the space spanned by $\phi(\boldsymbol{x}_i)$ does not contribute to the $\phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{x}')$ products, therefore it is sufficient to express $\phi(\boldsymbol{x})$ and $\boldsymbol{w}$ as a combination of $\phi(\boldsymbol{x}_i)$ vectors. The dimensionality d of the input vectors is frequently lower than the number of training patterns $d < m$, and then $\phi(\boldsymbol{x})$ represents mapping into higher m-dimensional space. According to the Cover theorem [11] probability of linear separation of data points grows with the dimensional of the space in which data is embedded. However, in case of microarray data and some other problems the reverse situation is true: dimensionality is much higher than the number of patterns for training. Still weighted distances used as features for linear discrimination may be quite useful, providing some improvement in comparison to the nearest neighbor methods.

The discriminant function in the $\phi()$ space is:

$$g(\boldsymbol{x}) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b \right) \tag{11}$$

If the kernel function is linear the $\phi()$ space is simply the original space and the contributions to the discriminant function are based on the cosine distances to the reference vectors $\boldsymbol{x}_i$ from the y_i class. Thus the original features $\boldsymbol{x}[j], j = 1..d$ are replaced by new features $z_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$ that evaluate how close (or how similar) the vector is from the training vectors. Incorporating signs in the coefficient vector $A_i = \alpha_i y_i$ discriminant functions is:

$$g(\boldsymbol{x}) = \text{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i z_i(\boldsymbol{x})) + b \right) = \text{sgn}\left(\boldsymbol{A} \cdot \boldsymbol{z}(\boldsymbol{x})) + b \right) \tag{12}$$

With the proper choice of non-zero α coefficients this function provides a distance measure from the decision border, combining distances from the support vectors placed at the margins. In non-separable case instead of using cosine distance measures it is better to use localized similarity measures, for example by scaling the distance with Gaussian functions. This leads to one of the most useful kernels:

$$k_G(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\beta \|\boldsymbol{x} - \boldsymbol{x}'\|^2) \tag{13}$$

Such kernels help to smooth decision borders because in the discriminant kernels anchored at the support vectors of one class are combined overcoming the influence of low-density data points from the opposite classes. Kernel-based methods use similarity in a special way in combination with linear discrimination, but similarity estimations may also be used in many other ways in pattern recognition problems [12, 1].

2.2 Kernel features spaces

For each vector $\boldsymbol{x}$ we have m kernel features $z_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$ defined for each training vector. For example taking the Gaussian kernel $k_G(\boldsymbol{x}, \boldsymbol{x}')$ and fixing the value of discriminant $g(\boldsymbol{x})$ =constant is equivalent to taking a weighted sum of gaussians centered at some support vectors that are near the border; for large dispersions all vectors may contribute, but a single one will have weak influence on the decision border. Because contours of discriminant function in the kernel space are approximately constant when $\boldsymbol{x}$ moves along the non-linear decision border in the input space, they lie on the hyperplane in the kernel space. Therefore in the space of kernel features linear discriminant methods may be applied directly, without the SVM machinery. This was demonstrated in computational experiments by comparing the results of SVM with Gaussian kernel solved by quadratic programming with direct linear solutions in the kernel-based feature space[3].

3 Locally Optimized Kernels

The Locally Optimized Kernels (LOK) approach presented in this paper is based on generation of new features using restricted gaussian kernels followed by the Winner Takes All (WTA) comparison of output activities (where a simple sum of the nodes assigned to class C is used to estimate the winning class), or by the linear discrimination. One may use many other machine learning algorithms in this new feature space [12, 1], but here we have space to compare only the basic version of this approach with standard SVM approaches (see Algorithm 1).

To create LOK feature space for every training vector a candidate kernel feature is used: $g_i(\boldsymbol{x}) = \exp(-||\boldsymbol{x}_i - \boldsymbol{x}||^2/2\sigma^2)$. For every such feature g_i analyze $p(g_i|C)$ distributions to find relatively pure clusters in some $I_{iab} = [g_{ia}, g_{ib}]$ interval (here we have used only pure clusters, although in some problems it may be necessary to include some. This creates binary candidate features $B_{iab}(\boldsymbol{x})$. Good candidate feature should cover some minimum number η of training vectors. The optimal number may depend on the type of data, the domain expert may consider even a single vector to be a significant exception worth retaining. Below the η parameter has been optimized using results

of crossvalidation. This condition avoids creation of overspecific features. In the implementation tested here LOK uses winner-takes-all mechanism or linear discrimination to find solutions in the new feature space.

The LOK algorithm is sketched below:

Algorithm 1 Locally Optimized Kernels

Require: Fix the values of internal parameters: η for minimum covering and σ for dispersion.

1: Standardize the dataset, m vectors, d features.
2: Create candidate kernel features $g_i(\boldsymbol{x}) = \exp(-||\boldsymbol{x}_i - \boldsymbol{x}||^2/2\sigma^2)$.
3: Sort $g_i(\boldsymbol{x})$ values in descending order, with associated class labels.
4: Analyze $p(g_i|C)$ distribution to find all intervals with pure clusters defining binary features $B_{iab}(\boldsymbol{x}; C)$.
5: **if** the number of vectors covered by the feature $B_{iab}(\boldsymbol{x}; C) > \eta$ **then**
6: accept this binary feature creating class-labeled hidden network node.
7: **end if**
8: Classify test data mapped into the enhanced space:
9: Sum the activity of hidden node subsets for each class to calculate network outputs (WTA).
10: Build linear model on the enhanced feature space (LDA).

In this version of LOK algorithm there are only two parameters to set: η determines the minimal size of a cluster (number of vectors per cluster), and σ controls dispersion of localized gaussian features. Clean clusters are found either in the local neighborhood of the support vector in the interval $[0, b]$, or if the support vector is surrounded by vectors from another class they may be quite far, with large values of both $a < b$. Thus even outliers may provide useful support features. Clean clusters and binary features may be quite useful to identify regions with vectors that may be correctly classified with high confidence. For very large datasets these vectors may be removed, leaving only areas close to the decision borders. In essence this solves the separable problem at a cost of high rejection rate. To deal with the remaining vectors one should introduce features based on clusters that are not pure.

For every candidate support vector point $b = g_i$ for which $p(g_i|C) = p(g_i|\neg C)$ is found and $\sigma_i = b/2$ is taken as dispersion, creating a new Gaussian kernel feature $g_i(\boldsymbol{x}; b) = \exp(-||\boldsymbol{x}_i - \boldsymbol{x}||^2/b)$. A slightly smaller value of b could make the new feature more pure, but this would introduce at least one additional parameter, therefore we have not considered this possibility. With sufficient number of support features small impurities for small $g_i(\boldsymbol{x})$ feature values do not matter. In some cases more support features may be generated using large σ_i and analyzing $p(g_i|C)$ distribution for values larger than b, using intervals $I_{iab} = [g_{ia}, gib], a > 0$ where one of the classes dominates. These new features are obtained as differences of two gaussian functions $g_i(\boldsymbol{x}; b) - g_i(\boldsymbol{x}; a)$. Other types of functions could be used here to model the slopes of probability density distributions, for example differences of two sigmoidal functions [13]. In the comparison of results presented in Table 2 LOKLDA and LOKWTA are used with such additional features, if they were found useful improving the training results.

To find solution in the new feature space LOKLDA uses linear discrimination with margin maximization (optimal margin is selected using crossvalidation in a standard way, as it is done also for SVM calculations). Features discovered by the LOK algorithm may be implemented as network nodes that represent kernel transformations. Additional layers of the network are then used to analyze the data in the feature space created in this way (see Fig. 1). LOK algorithm with Gaussian features may be called properly LOGK, as other functions may be used as kernels.

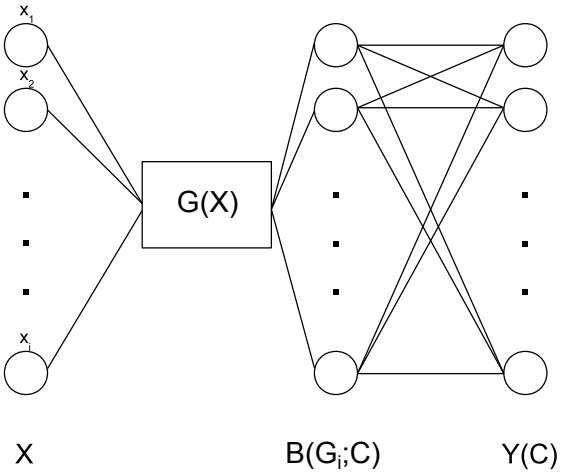

Fig. 1. Network structure of the LOK algorithm.

4 Illustrative examples

In order to evaluate the effect of optimized kernels on SVM results 4 methods have been compared: standard SVM with linear kernel (SVML) and with Gaussian kernel (SVMG), LOK with the Winner Takes All (WTA) estimation (LOKWTA) and LOK with linear discrimination (LOKLDA), equivalent to the linear SVM in the extended space. These 4 approaches have been applied to the 27 standard benchmark datasets, summarized in Table 1, downloaded from the UCI Machine Learning Repository [14]. Vectors with missing feature values (if any) have been removed. Each calculation has been performed using the 10-fold stratified crossvalidation, and repeated 10 times to obtain reliable estimates of accuracies and standard deviations. The SVM parameters C (best value from the $[2^{-5} \ldots 2^{5}]$ range) and σ (best value from the $[2^{-10} \ldots 2^{3}]$ range) have been fully optimized for each dataset in an automatic way, using crossvalidation estimations on the training partition to be sure that no information about the test data has been used at any stage. Results for each dataset are collected in Table 2, with the best results marked in bold.

Results using Locally Optimized Kernels are in all cases not significantly worse than SVM, and in most cases better. As should be expected LOKWTA achieved best

Table 1. Summary of datasets

Dataset	#Vectors	#Features	#Classes	Dataset	#Vectors	#Features	#Classes
arrhythmia	63	279	11	autos	159	25	6
balance-scale	625	4	3	breast-cancer	277	9	2
breast-w	683	9	2	car	1728	6	4
cmc	1473	9	3	credit-a	653	15	2
credit-g	1000	20	2	cylinder-bands	277	39	2
dermatology	358	34	6	diabetes	768	8	2
ecoli	336	7	8	glass	214	9	6
haberman	306	3	2	heart-c	296	13	2
heart-statlog	270	13	2	hepatitis	80	19	2
ionosphere	351	34	2	iris	150	4	3
kr-vs-kp	3196	36	2	liver-disorders	345	6	2
lymph	148	18	4	sonar	208	60	2
vote	232	16	2	vowel	990	13	11
				zoo	101	17	7

results only in a few (exactly 4) cases, in case of *cmc* data outperforming SVM by a large margin. SVML also achieved best results in 4 cases, in case of *arrhythmia* (only 63 vectors with 11 classes and 279 features) strongly outperforming other methods, but in this case variance is quite big and the difference is not statistically significant.

For 11 datasets LOKLDA outperformed all other methods, while SVMG was the best for 8 datasets, and only in the case of *car* data the difference becomes significant. These results show that in most cases local optimization of kernels leads to an improvement over the single kernel SVM algorithms, and may achieve the best results comparing to all state-of-the-art classifiers (Maszczyk, PhD thesis, in prep.). The computational complexity of the LOKLDA approach is dominated by solution of linear discrimination problem and thus is comparable to the original linear SVM. For d-dimensional problems and m vectors estimation of optimal kernel size requires $O(dm \log m)$ operations. This is the computational complexity of the LOKWTA approach that in many cases has not been significantly worse than SVM, while the method may be used with very large datasets, where solving linear discrimination becomes costly.

5 Conclusions

Locally Optimized Kernels (LOK) algorithm introduced in this paper is focused on generation of new useful kernel features. It is not restricted to gaussian kernels, and may be treated as one variant of our general Support Feature Machine approach [3] for generation of features that extract more information from the data than may be derived using simple kernels. The purpose of this paper is to show that the LOK approach despite its simplicity generates enhanced feature space that improves construction of the original SVM kernel space. While a lot of effort has been devoted to improvements of the SVM learning algorithm much less attention has been paid to methods that extract information from data, making the linear discrimination simpler and more accurate. In

Table 2. 10x10CV results

Dataset	SVML	SVMG	LOKWTA	LOKLDA
arrhythmia	**50.92±17.31**	43.36±21.47	42.00±24.19	39.10±12.98
autos	54.48±13.75	74.29±12.58	58.69±11.03	**74.36±10.40**
balance-scale	84.47±3.17	89.83±2.09	90.71±2.38	**96.46±2.62**
breast-cancer	73.27±6.10	75.67±5.35	**76.58±6.37**	75.09±1.99
breast-w	96.60±2.07	96.77±1.84	96.93±1.62	**97.21±2.13**
car	67.99±2.61	**98.90±0.90**	84.72±3.44	93.57±1.81
cmc	19.14±2.14	34.09±3.67	48.54±2.52	**51.06±4.30**
credit-a	**86.36±2.86**	86.21±2.90	82.67±4.01	84.70±4.91
credit-g	73.95±4.69	**74.72±4.03**	73.10±2.38	72.70±3.86
cylinder-bands	74.58±5.23	76.89±7.57	74.32±6.41	**80.11±7.53**
dermatology	94.01±3.54	94.49±3.88	87.97±5.64	**94.71±3.02**
diabetes	76.88±4.94	76.41±4.22	74.88±3.88	**76.95±4.47**
ecoli	78.48±5.90	84.17±5.82	82.47±3.66	**85.66±5.40**
glass	42.61±10.05	62.43±8.70	64.96±7.72	**71.08±8.13**
haberman	72.54±1.96	72.91±5.93	**76.46±4.34**	73.53±0.72
heart-c	**82.62±6.36**	80.67±7.96	81.07±7.56	81.04±5.17
heart-statlog	**83.48±7.17**	83.40±6.56	81.48±8.73	83.33±7.46
hepatitis	83.25±11.54	84.87±11.98	**89.88±10.14**	84.05±4.40
ionosphere	87.72±4.63	94.61±3.68	85.18±6.28	**95.16±2.72**
iris	72.20±7.59	**94.86±5.75**	94.67±6.89	93.33±5.46
kr-vs-kp	96.03±0.86	**99.35±0.42**	83.73±2.58	98.25±0.45
liver-disorders	68.46±7.36	**70.30±7.90**	57.40±5.72	69.72±6.57
lymph	81.26±9.79	**83.61±9.82**	76.96±13.07	80.52±7.91
sonar	73.71±9.62	86.42±7.65	**86.57±7.01**	86.52±8.39
vote	96.12±3.85	**96.89±3.11**	92.57±7.52	93.95±4.18
vowel	23.73±3.13	**98.05±1.90**	92.49±3.37	97.58±1.52
zoo	91.61±6.67	93.27±7.53	88.47±5.35	**94.07±6.97**

the case of gaussian kernel features there is no reason why the same dispersion should be used for all support vectors. Those support vectors that are far from decision border on the correct side should have large dispersions, and vectors closer to decision borders should have smaller dispersions. Support vectors on the wrong side should provide kernel features that exclude local neighborhood. LOK algorithm creates such locally optimized gaussian kernels. It may also be easily combined with many methods of feature selection for additional improvement.

A fruitful question is: what is the limit of accuracy for a given dataset that can be achieved in a given feature space? Progress in the recent years in classification and approximation methods shows that for simple data results are probably close to this limit. However, there is still an ample room for improvement in generation of enhanced features spaces that contain more information. Of course not only kernel-based features are useful. Several ways of creating new features have recently been introduced [15, 3, 16], overcoming the limitation of algorithms that work in original features spaces, where discrimination or similarity-based methods operate. The final goal is to create

meta-learning approaches [17] that construct in automatical way optimal, and in most cases the simplest, models of data.

Acknowledgment: This work was supported by the Nicolaus Copernicus University under research grant for young scientists no. 407-F, and by the Polish Ministry of Education and Science through Grant No N516 500539.

References

1. Duch, W.: Similarity based methods: a general framework for classification, approximation and association. Control and Cybernetics **29** (2000) 937–968
2. Duch, W.: k-separability. Lecture Notes in Computer Science **4131** (2006) 188–197
3. Maszczyk, T., Duch, W.: Support feature machines: Support vectors are not enough. In: World Congress on Computational Intelligence, IEEE Press (2010) 3852–3859
4. Tebbens, J., Schlesinger, P.: Improving implementation of linear discriminant analysis for the small sample size problem. Computational Statistics & Data Analysis **52** (2007) 423–437
5. Duch, W., Maszczyk, T.: Almost random projection machine. Lecture Notes in Computer Science **5768** (2009) 789–798
6. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: In D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA, ACM Press. (1992)
7. Schölkopf, B., Smola, A.: Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA (2001)
8. Schölkopf, B., Burges, C., Smola, A.: Advances in Kernel Methods: Support Vector Machines. MIT Press, Cambridge, MA (1998)
9. Diederich, J., ed.: Rule Extraction from Support Vector Machines. Volume 80 of Springer Studies in Computational Intelligence. Springer (2008)
10. Chapelle, O.: Training a support vector machine in the primal. Neural Computation **19** (2007) 1155–1178
11. Cover, T.M.: Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Transactions on Electronic Computers **14** (1965) 326—-334
12. Duch, W., Adamczak, R., Diercksen, G.: Classification, association and pattern completion using neural similarity based methods. Applied Mathematics and Computer Science **10** (2000) 101–120
13. Duch, W., Jankowski, N.: Survey of neural transfer functions. Neural Computing Surveys **2** (1999) 163–213
14. Asuncion, A., Newman, D.: UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html (2007)
15. Duch, W., Maszczyk, T.: Universal learning machines. Lecture Notes in Computer Science **5864** (2009) 206–215
16. Maszczyk, T., Grochowski, M., Duch, W. In: Discovering Data Structures using Meta-learning, Visualization and Constructive Neural Networks. Volume 262 of Advances in Machine Learning II. Springer Series: Studies in Computational Intelligence, Vol. 262 (2010) 467–484
17. Jankowski N, Duch W, G.K., ed.: Meta-learning in Computational Intelligence. Volume 358 of Studies in Computational Intelligence. Springer (2011)