Lecture Notes in Artificial Intelligence Vol. 6086, pp. 178-186, 2010.

Support Feature Machine for DNA microarray data.

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 Feature Machines (SFM) define useful features derived from similarity to support vectors (kernel transformations), global projections (linear or perceptron-style) and localized projections. Explicit construction of extended feature spaces enables control over selection of features, complexity control and allows final analysis by any classification method. Additionally projections of high-dimensional data may be used to estimate and display confidence of predictions. This approach has been applied to the DNA microarray data.

Key words: Features generation, dimensionality reduction, learning, support feature machines, support vector machines

1 Introduction

Data mining packages such as Weka [1], Rapid Miner [2], or KNIME [3] offer enormous number of modules for pre-processing, optimization, training, classification, clustering, visualization and post-processing. For example, combining all modules available in Rapid Miner 3.4 over 5 billion data models may be created. Comparison of all these models on a single dataset would be impossible and will anyway manifest results of "oversearching", with some models producing good results on a given dataset by pure chance [4]: contrary to the common opinion generalization of learning systems (including decision trees and neural networks) trained using global optimization methods (such as evolutionary algorithms) is frequently worse than results obtained from the greedy, best-first methods (such as the gradient methods). Meta-learning, or creating on demand optimal adaptive system suitable for a given problem is an answer to this crises of abundance [5, 6]. Different approaches to meta-learning have been proposed [7], based either on recommendations of particular learning methods depending on some data characteristics (landmarking approach), combining many data models together [8], or defining frameworks that allow for systematic creation of data models of growing complexity. In particular, a framework based on evaluation of similarity [5], that includes many variants of the nearest neighbor methods, neural networks and kernel methods [9], has proved to be very powerful in practical applications. Within such framework methods that have a proper bias for particular problem may be found.

Each data model M_i is defined in some hypotheses space $\mathcal{H}$ that includes all functions that this model may learn. Combination of diverse classifiers $p(C|\boldsymbol{x}, M_i)$ that predict probabilities or provide binary indicators for each class C, may improve results.

Mixture of experts [8], and in particular the boosted mixtures of experts [10], assign large weights to classifiers (called experts) that improve performance around localized areas of the input space where most models fail. Committees of competent models [11, 12] have weights $\boldsymbol{w}_i(\boldsymbol{x})p(C|\boldsymbol{x}, M_i)$ that explicitly depend on the input vectors $\boldsymbol{x}$, decreasing the influence of classifiers that have not been competent, or had low confidence in their predictions in some regions of the input space. Stacking classifiers is another technique that trains to predict errors that the current set of classifiers makes [13].

Committees and mixture of experts are based on component models optimized for the whole training data. All these methods decrease comprehensibility of solutions and are not the most efficient way of summarizing the data. Recently we became interested in ensembles of simple models at finer granulation, defining classifiers based on single feature and provide data models that are competent only for relatively small subsets of all samples. The most common Gaussian kernel Support Vector Machine (SVM) [9] selects a subset of training vectors $\boldsymbol{x}_i$ close to the decision border (called "support vectors") and calculates $k(\boldsymbol{x}, \boldsymbol{x}_i) = \exp(-\beta \sum |\boldsymbol{x}_i - \boldsymbol{x}|^2)$, with fixed dispersion β. The final discriminant function is constructed as a linear combination of such kernels, creating in fact a weighted nearest neighbor solution. Other similarity estimation based on specific kernels (similarity functions) may help to increase flexibility of decision borders, decreasing the number of kernels need for accurate classification.

Each support vector used in a kernel may provide a useful feature, but this type of solution is optimal only for data with particular distributions, and will not work well for example on parity data [14] or other problems with complex logical structure [15]. For some highly-non-separable problems localized linear projections may easily solve the problem [16]. Adding new types of features extends the hypothesis space. Useful features may be created by random linear projections [17], or principal components derived from data, or projection pursuit algorithms based on Quality of Projected Clusters (QPC) indices [18]. Non-linear combinations of input features provided by neural network nodes may also be used.

Creation of appropriate feature space facilitates finding optimal solutions, and thus is worthwhile exploring not only at the level of combination of classifiers, but in the first place to learn from other models what interesting features they have discovered. They may come in form of prototypes, linear combinations, or fragments of branches in decision trees, forming useful “knowledge granules” in data, as it is done in our Universal Learning Machines [19]. The final model – linear discrimination, Naive Bayes, nearest neighbor or a decision tree – is secondary, if appropriate space has been set up.

In the next section SFM algorithm as used here is introduced, and in section 3 tested on several microarray data. Brief discussion of further research directions concludes this paper.

2 Support Feature Machine Algorithm

Support Vector Machines [20, 9] used as classifiers are based on linear discrimination, with maximization of classification margin that ensures good generalization and allows for control of complexity. Problems that require non-linear decision borders may be linearized by projecting the data into high-dimensional feature space. According to the

Cover theorem [21] such mapping increases the probability of making the data separable, "flattening" decision borders. Kernel methods implicitly provide new features $z_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$ constructed around support vectors $\boldsymbol{x}_i$, selected from the training close to the decision borders. The number of these features is equal to the number of training vectors n. The ability to solve classification problem using only the kernel matrix $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ has some advantages, as instead of the original input space $\boldsymbol{x}_i$ one works in the space of kernel features $z_i(\boldsymbol{x})$,called further "the kernel space".

The number of input features N may be smaller or greater than n. If $N \ll n$ the number of the kernel features is usually unnecessarily large – think of the simple case of two Gaussian distributions that are optimally solved by forming a projection connecting their means, instead of a large number of support vectors with Gaussian kernels close to the decision border. Adding such linear projections to the list of features, and performing feature selection may in such situation remove most kernel features as irrelevant. In the reverse situation, when $n \ll N$ (as is the case for the microarray data), instead of a projection into high-dimensional space SVM reduces the number of features to no more than n. Also in this case other types of features may be useful. In particular original input features may be kept in the enhanced feature space (SVM does not use them), or various projections may be added. This should guarantee that simplest solutions to easy problems are not overlooked (SVM will miss them, even if a single binary feature is sufficient).

Support Feature Machines used here generalize SVM approach by explicitly building enhanced space that includes kernel features $z_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x}_i)$ together with any other features that may provide useful information. This approach has several advantages comparing to standard SVM:

1. With explicit representation of features interpretation of discriminant function is as simple as in any linear discrimination method.
2. Kernel-based SVM is equivalent to linear SVM in the explicitly constructed kernel space, therefore enhancing this space should lead to improvement of results.
3. Kernels with various parameters may be used, including various degrees of localization, and the resulting discriminant may select global features, combining them with local features that handle exceptions (SVM algorithms are usually formulated using single kernel, with a few exceptions [22]).
4. Complexity of SVM is $O(n^2)$ due to the need of generating kernel matrix; SFM may select smaller number of kernel features at $O(n)$ cost from those vectors that project on overlapping regions in linear projections.
5. Many feature selection methods may be used to estimate usefulness of new features that define support feature space.
6. Many algorithms may be used in the support feature space to generate the final solution.

Although one can use various methods to solve classification problem in the space created by support features, linear SVM approach is used here for several reasons. First, there are many excellent and well-tested SVM implementations available. Second, linear SVM includes regularization term, ensuring large margin solution. For microarray

data other linear techniques designed for small sample problems may be more appropriate [23] and will be tried in future. We shall consider other machine learning algorithms in future SFM versions.

In this paper only basic version of this approach is implemented (see Algorithm 1) using linear discrimination to obtain final decision function. SFM algorithm starts from standardization of the whole data, followed by feature selection, based here on the Relief algorithm [24], leaving only features with positive weights. Such reduced, but still high dimensional data, is used to generate two types of new features.

First type of features is made by projections on $m = N_c(N_c - 1)/2$ directions obtained by connecting pairs of centers $\boldsymbol{w}_{ij} = \boldsymbol{c}_i - \boldsymbol{c}_j$, where $\boldsymbol{c}_i$ is the mean of all vectors that belong to the $C_i, i = 1...N_c$ class. In high dimensional space such features $r_i(\boldsymbol{x}) = \boldsymbol{w}_i \cdot \boldsymbol{x}$ help a lot, as can be seen in Fig. 1, where smoothed histograms of data projected on such directions are shown. Fisher discriminant analysis may provide even better directions here, but this is the least expensive solution, always worth adding to the pool of new features.

The second type are features based on similarity to the training vectors, or kernel features. While many types of kernels may be mixed together, including the same types of kernels with different parameters, in the initial implementation only Gaussian kernels with fixed dispersion β are taken for each training vector $t_i(\boldsymbol{x}) = \exp(-\beta \sum |\boldsymbol{x}_i - \boldsymbol{x}|^2)$. Various ways of selecting suitable training vectors may be considered, but for small sample data all of them may be taken into account. QPC algorithm [18] has been used on this feature space, generating additional orthogonal directions that are useful for visualization and as new features. The number of QPC features has been arbitrarily set to 5, although optimizing this parameter in crossvalidation should give better final result.

Algorithm 1 Algorithm

Require: Fix the Gaussian dispersion β and the number of QPC features N_Q.

1: Standardize dataset.
2: Normalize the length of each vector to 1.
3: Perform Relief feature ranking, select only those with positive weights $RW_i > 0$.
4: Calculate class centers $\boldsymbol{c}_i, i = 1...N_c$, create m directions $\boldsymbol{w}_{ij} = \boldsymbol{c}_i - vc_j, i > j$.
5: Project all vectors on these directions $r_{ij}(\boldsymbol{x}) = \boldsymbol{w}_{ij} \cdot \boldsymbol{x}$ (features r_{ij}).
6: Create kernel features $t_i(\boldsymbol{x}) = \exp(-\beta \sum |\boldsymbol{x}_i - \boldsymbol{x}|^2)$.
7: Create N_Q QPC directions $\boldsymbol{w}_i$ in the kernel space, adding QPC features $s_i(\boldsymbol{x}) = \boldsymbol{w}_i \cdot \boldsymbol{x}$.
8: Build linear model on the new feature space.
9: Classify test data mapped into the new feature space.

In essence SFM algorithm requires construction of new features, followed by a simple linear model (linear SVM has been used here) or any other learning model. More attention is paid to generation of features than to the sophisticated optimization algorithms or new classification methods. Although several parameters may be used to control the process of feature creation and selection they are either fixed or set in an automatic way. Solutions are given in form of linear discriminant function and thus are

easy to understand. New features created in this way are based on those transformations of inputs that have been found interesting for some task, and thus have meaningful interpretation (see Fig. 1, 2. The importance of generating new features has already been stressed in our earlier papers, but they have been based either on random projections [17], extracted from several types of algorithms such as decision trees or the nearest-neighbor methods [19], or provided by classification algorithms [25]. Systematic creation of support feature spaces as described in this paper seems to be an unexplored approach with great potential.

3 Illustrative examples

The usefulness of SFM approach has been tested on several DNA microarray datasets and compared with SVM and SSV[26] results. These high-dimensional datasets have been used in the Discovery Challenge at the 7-th International Conference on Rough Sets and Current Trends in Computing (RSCTC 2010), Warsaw, Poland (28-30 June, 2010). A summary of these datasets is presented in Tab. 1.

Title	#Features	#Samples	Class distribution
Data 1	54675	123	88–35
Data 2	22283	105	40–58–7
Data 3	22277	95	23–5–27–20–20
Data 4	54675	113	11–31–51–10–10
Data 5	54613	89	16–10–43–20
Data 6	59004	92	11–7–14–53–7

Table 1. Summary of used datasets.

Because the number of samples in some classes is very small, in the k-fold cross-validation tests performed for evaluation of SFM performance, k is equal at least to the number of vectors in the smallest class (from 5-10), and at most 10. Average results are collected in Table 2, with accuracy, balanced accuracy (accuracy for each class) and standard deviation given for each dataset. Additionally values of parameters determined in internal crossvalidation are given. Number of features after the relief selection is still very large and it is clear that more conservative selection could be used; these features are used only to generate a small number of r, t and s-type of features. Number of new features NF includes 5 QPC and $N_c(N_c-1)/2$ directions connecting centers, so together with the kernel feature the support feature space has at most 129 dimensions.

Optimal dispersion of kernel features is in most cases quite large, up to 2^8, creating very smooth, partially almost linear decision border in the kernel space. These results show by no means the limits of SFM algorithm, as fixed parameters have been used in all tests, more features could be generated using random projections, alternative kernels, more QPC directions, and other linear discrimination algorithms or other machine learning algorithms should be tested to generate final decision functions. Unfortunately exploration of all these possibilities is quite time consuming and software to perform all necessary comparisons in an automatic way is under development.

The microarray data in particular require small sample methods with appropriate smoothing. This is not the best domain to evaluate learning algorithms, as drawing

Table 2. Accuracies (ACC) and balanced accuracies (BACC) for each used dataset. Also for SFM optimal parameter β is noted, number of features after relief selection (FAS), number of new features (NF), and number of folds used in crossvalidation tests.

Dataset	Info			SVM		SSV		SFM		
	FAS	NF	CV	ACC	BACC	ACC	BACC	ACC	BACC	β
Data 1	51259	117	10	93.5±6.3	90.5±9.7	74.0±13.0	71.0±17.0	91.5±9.3	89.9±11.5	2^3
Data 2	14880	98	7	60.9±10.5	54.2±18.3	56.2±8.4	35.4±6.3	67.8±15.5	67.1±14.3	2^8
Data 3	15628	91	5	75.8±7.9	65.9±6.1	74.7±4.4	66.6±4.3	96.0±5.4	89.3±8.9	2^3
Data 4	7529	116	10	38.1±9.9	28.9±8.9	38.8±12.2	19.8±8.1	54.1±15.8	41.5±13.1	2^{-4}
Data 5	31472	91	10	60.4±18.4	55.1±21.7	59.4±12.3	49.0±21.3	68.6±7.9	64.9±9.6	2^3
Data 6	47307	88	7	61.9±8.5	46.4±13.4	57.5±2.6	24.1±8.2	79.4±17.2	53.1±15.5	2^0

conclusions from balanced accuracy on data with a few such samples per class in a space of such huge dimension without any domain knowledge is impossible. However, this is quite challenging data and therefore a comparison of SFM with SVM results is interesting.

4 Discussion and new directions

Support Feature Machine algorithm introduced in this paper is focused on generation of new features, rather than improvement of optimization and classification algorithms. It may be regarded as an example of mixture of experts, where each expert is a simple model based on projection on some specific direction (random, or connecting clusters), localization of projected clusters (QPC), optimized directions (for example by Fisher discriminant analysis), or kernel methods based on similarity to reference vectors. Obviously there is a lot of room for improvement. For some data kernel-based features are most important, for other projections and restricted projections discover more interesting aspects. Recently more sophisticated ways of creating new features have also been introduced [19, 25], deriving new features from various data models. For example, combination of several features with appropriate thresholds obtained from decision trees, create interesting semi-local features.

Instead of simple linear combination competent committee may be introduced. Calculating probability $p(C_i|\boldsymbol{x}; M)$ of assigning $\boldsymbol{x}$ vector to class C_i by the final model M, given probabilities for each feature $P(C_i|\boldsymbol{x}; M_j)$ (as shown in Fig. 1), coefficients of linear combination are determined from the least-mean square solution of:

$$p(C_i|\boldsymbol{x}; M) = \sum_{j=1}^{m} \sum_{m} W_{ij} F(\boldsymbol{x}_j) P(C_i|\boldsymbol{x}_j) \tag{1}$$

where the incompetence factors $F(\boldsymbol{x}_j)$ are estimated from histograms $P(C_i|\boldsymbol{x}_j)$. These factors simply modify probabilities $F(\boldsymbol{x}; M_j)P(C_i|\boldsymbol{x}_j)$ that are used to set up models in the enhanced feature space. This approach requires only minimal change in the whole algorithm. After renormalization $p(C_i|\boldsymbol{x}; M)/\sum_j P(C_j|\boldsymbol{x}; M)$ gives final probability

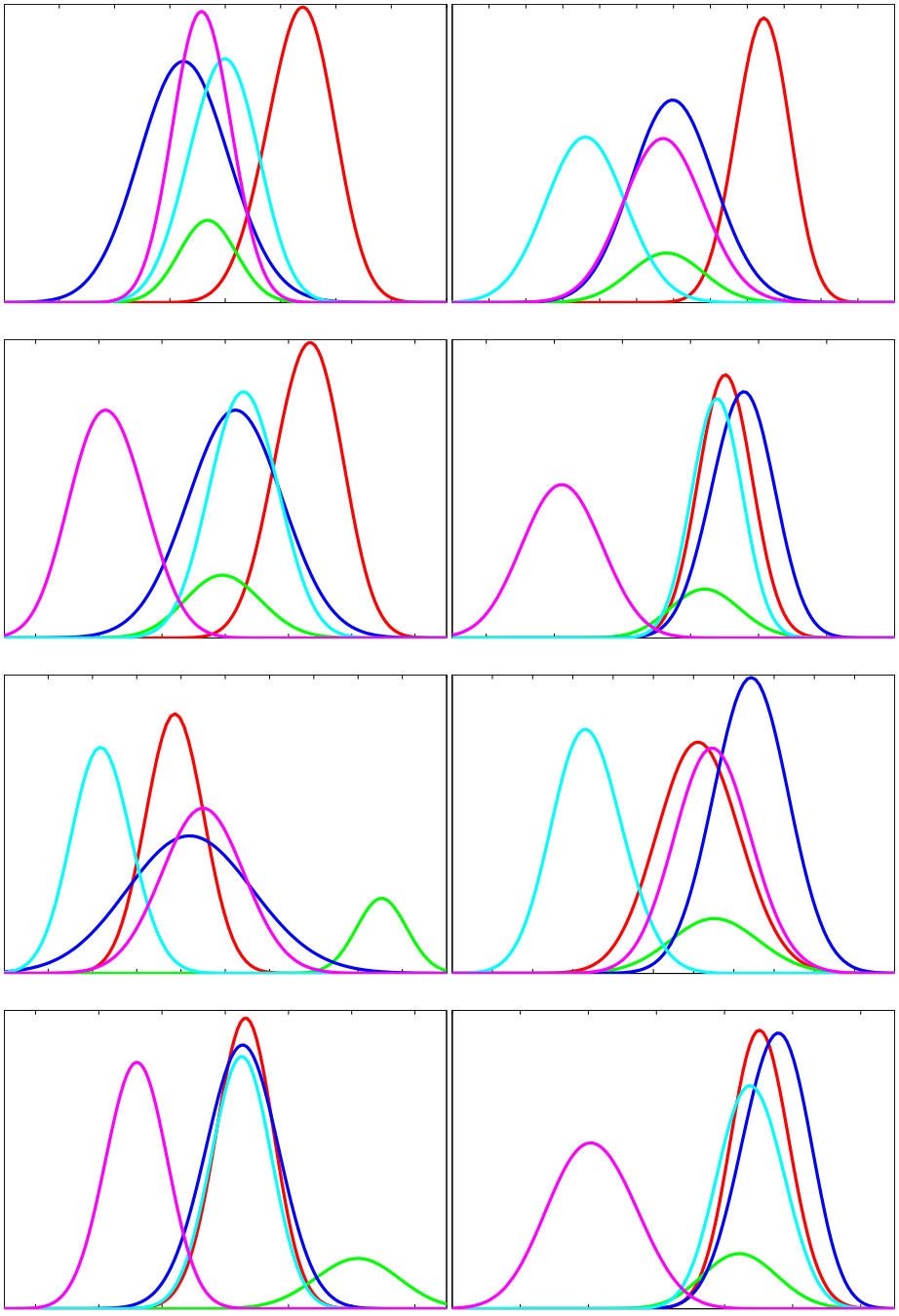

Fig. 1. Histograms for 5 classes (Dataset 3), two QPC projections on lines connecting centers of these classes.

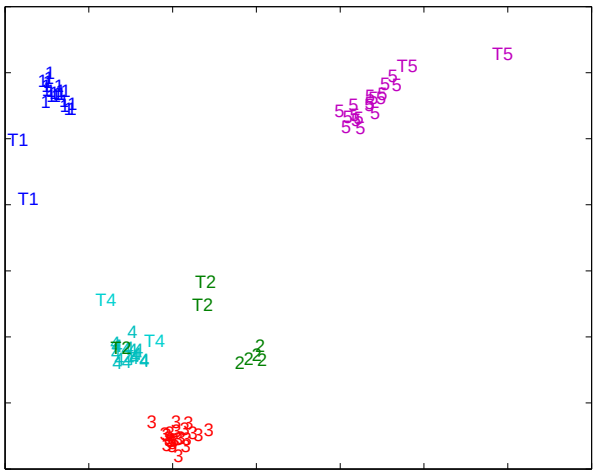

Fig. 2. Scatterogram showing distribution of training and test vectors in QPC directions derived from kernel features, taken from one of the 10-fold crossvalidation folds.

of classification. In contrast to AdaBoost and similar procedures [13] explicit information about regions of incompetence, or quality of classifier based on single new feature in different feature space areas, is used. Many other variants of basic SFM algorithm reported here are possible. It is worth noting that kernel-based SVM is equivalent to the use of kernel features combined with linear SVM. Mixing different kernels and different types of features creates much better enhanced features space then a single-kernel solution. For example, complex data may require decision borders of different complexity, and it is rather straightforward to introduce multiresolution in the presented algorithm, for example using different dispersion β for every $\boldsymbol{t}_i$, while in the standard SVM approach this is difficult to achieve.

References

1. Witten, I., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2nd Ed, 2005)
2. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: Proc. 12th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2006). (2006)
3. Berthold, M., Cebron, N., Dill, F., Gabriel, T., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., Wiswedel, B.: KNIME: The Konstanz Information Miner. In: Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007), Springer (2007)

4. Quinlan, J., Cameron-Jones, R.: Oversearching and layered search in empirical learning. In: In Proc. of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1995) 1019–1024
5. Duch, W., Grudziński, K.: Meta-learning via search combined with parameter optimization. In Rutkowski, L., Kacprzyk, J., eds.: Advances in Soft Computing. Physica Verlag, Springer, New York (2002) 13–22
6. Grabczewski, K., Jankowski, N.: Meta-learning with machine generators and complexity controlled exploration. Lecture Notes in Artificial Intelligence **5097** (2008) 545–555
7. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Cognitive Technologies. Springer (2009)
8. Kuncheva, L.: Combining Pattern Classifiers. Methods and Algorithms. J. Wiley & Sons, New York (2004)
9. Schölkopf, B., Smola, A.: Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA (2001)
10. Avnimelech, R., Intrator, N.: Boosted mixture of experts: An ensemble learning scheme. Neural Computation **11** (1999) 483–497
11. Duch, W., Itert, L.: Committees of undemocratic competent models. In Rutkowski, L., Kacprzyk, J., eds.: Proc. of Int. Conf. on Artificial Neural Networks (ICANN), Istanbul. (2003) 33–36
12. Duch, W., Itert, L.: Competent undemocratic committees. In Rutkowski, L., Kacprzyk, J., eds.: Neural Networks and Soft Computing. Physica Verlag, Springer (2002) 412–417
13. Smyth, P., Wolpert, D.: Linearly combining density estimators via stacking. Machine Learning **36** (1999) 59–83
14. Brown, D.: N-bit parity networks. Neural Networks **6** (1993) 607–608
15. Grochowski, M., Duch, W.: Learning highly non-separable Boolean functions using Constructive Feedforward Neural Network. Lecture Notes in Computer Science **4668** (2007) 180–189
16. Duch, W.: k-separability. Lecture Notes in Computer Science **4131** (2006) 188–197
17. Duch, W., Maszczyk, T.: Almost random projection machine. Lecture Notes in Computer Science **5768** (2009) 789–798
18. Grochowski, M., Duch, W.: Projection Pursuit Constructive Neural Networks Based on Quality of Projected Clusters. Lecture Notes in Computer Science **5164** (2008) 754–762
19. Duch, W., Maszczyk, T.: Universal learning machines. Lecture Notes in Computer Science **5864** (2009) 206–215
20. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other Kernel-Based Learning Methods. Cambridge University Press (2000)
21. 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
22. Sonnenburg, S., G.Raetsch, C.Schaefer, B.Schoelkopf: Large scale multiple kernel learning. Journal of Machine Learning Research **7** (2006) 1531–1565
23. Tebbens, J., Schlesinger, P.: Improving implementation of linear discriminant analysis for the small sample size problem. Computational Statistics & Data Analysis **52** (2007) 423–437
24. Robnik-Sikonja, M., Kononenko, I.: Theoretical and empirical analysis of relieff and rrelieff. Machine Learning **53** (2003) 23–69
25. Maszczyk, T., Grochowski, M., Duch, W.: Discovering Data Structures using Meta-learning, Visualization and Constructive Neural Networks. In: Advances in Machine Learning II. Volume 262. Springer Series: Studies in Computational Intelligence (2010) 467–484
26. Grąbczewski, K., Duch, W.: The separability of split value criterion. In: Proceedings of the 5th Conf. on Neural Networks and Soft Computing, Zakopane, Poland, Polish Neural Network Society (2000) 201–208