Universal Learning Machines

Włodzisław Duch and Tomasz Maszczyk

Department of Informatics, Nicolaus Copernicus University, Toruń, Poland
Google: W Duch
tmaszczyk@is.umk.pl

Abstract. All existing learning methods have particular bias that makes them suitable for specific kind of problems. Universal Learning Machine (ULM) should find the simplest data model for arbitrary data distributions. Several ways to create ULMs are outlined, and an algorithm based on creation of new global and local features combined with meta-learning is introduced. This algorithm is able to find simple solutions that sophisticated algorithms ignore, learn complex Boolean functions, complicated probability distributions, as well as the problems requiring multiresolution decision borders.

1 Introduction

Despite great progress in development of numerous new algorithms computational intelligence (CI) systems are still far behind natural biological systems in solving complex problems that face organisms, learning to optimize chances of survival, recognizing important structures in perceptual information, developing communication and coordination with other organisms. CI algorithms are quite sophisticated, but the key to general intelligence may lie in specific information filters that make learning possible, and chunking mechanisms that combine their results into higher-level mental representations. Filters discover phonemes, syllables, words in the auditory stream (with even more complex hierarchy in the visual stream), while chunking links sequences of lower level patterns into higher-level patterns, discovering associations and motifs.

On a more technical level this means that more attention should be paid to generation of features, exploiting various ways to use input information. Systematic selection and construction of new features should be followed by simple learning models that are quite accurate once an appropriate representation of the problem is found. For highly non-linear decision borders local features based on kernels localized at data samples that are close to decision borders are useful. There is some similarity to the kernel-based learning [1] that implicitly projects data into high-dimensional spaces, but here mixing different kernels and using different types of features gives much more flexibility. Cortical minicolumns extract a large number of interesting features from signals. This inspiration led to almost Random Projection Machine (aRPM) [2] algorithm where random projections generate new candidate features, and only those that contain clusters of pure data samples (in the k-separability sense [3]) are selected. The algorithm is fast and works quite well for complex Boole'an functions where almost all other algorithms fail. However, for some data distributions direct use of linear projections is not the best idea. For example, visual object recognition should distinguish complex shapes

C.S. Leung, M. Lee, and J.H. Chan (Eds.): ICONIP 2009, Part II, LNCS 5864, pp. 206–215, 2009.

defined in two- or three-dimensional space. Sophisticated approaches may miss simple and comprehensible solutions.

In this paper Universal Learning Machines (ULM) are introduced, based on systematic generation and selection of features of growing complexity, that help to learn simple data models in all kinds of situations. This is the next step towards meta-learning systems [4,5], that shifts emphasis on construction of the feature space, rather than construction of sophisticated learning models. In the following section short description of meta-learning idea is given, followed by systematic description of feature constructors and some details about current implementation of the ULM approach. Illustrative results are in section 4 and a brief discussion concludes this paper.

2 Meta-Learning and Feature Construction

According to the "no free lunch" theorem [6] no single system may reach the best results for all possible distributions of data. Decision trees and rule-based systems are the best for data that have simple logical structure, require sharp decision borders [7,8] but fail already on problems where linear discrimination provides accurate solution. SVM in kernelized form works well when complex topology is required but may miss simple solutions that rule-based systems find, fails when sharp decision borders are needed and fails on complex Boolean problems [9]. Each system has a particular bias that makes it suitable for particular class of problems. Discovering this bias and finding an appropriate model is not easy, and is usually done by tedious experimentations with combinations of pre-processing, filtering and selection, clusterization, classification or regression and post-processing techniques, combined with meta-learning procedures based on stacking, boosting, committees and other techniques. A number of efforts directed at automatic search for good data models are worth noting.

First approach is to search for general characteristics of data and compare it to the characteristics of reference data for which ranking of results of different system has been established [10]. However, a small change in data distribution may have a strong influence on the type of decision borders needed. The second approach tries to put various data transformations into a framework that systematically increases their complexity and do a meta-search in the space of all such models. The framework for similarity-based methods is quite suitable here, covering most classification methods, including feedforward neural networks and some kernel methods [4,5]. The granularity of the transformations use to automatically build good models is finer when heterogeneous systems are admitted, using optimized distance (kernel) functions of different types and optimized neural transfer functions in constructive network methods [11,12]. Other systems may be added to the pool of models, including heterogeneous decision trees and their forests [13]. This approach may be converted into general theory of transformation based learning, building models based on composition of transformations [14], and combining them by using committees of locally-competent models [15].

With proper control of search and complexity of generated models [16,17] such transformation-based approach offers an interesting approach that may overcome the limits of the "no free-lunch" theorem. However, success of such meta-search for best composition of transformations relies on the availability of transformations that extract

useful features and handle various specific problems, such as image analysis, multimedia streams, signal decomposition, text analysis, biosequences and many other problems. A Universal Learning Machine (ULM) is based on a very general algorithm:

- Construct new features (generate transformations).
- Test if they help to learn (generate results and select).

Learning may of course be done with one of the sophisticated schemes, but here the focus will be on construction of features. They may be ranked by some simple filter criterion [18], or be used to incrementally expand feature space using simple learning models. Analysis of all feature constructors for specific applications is beyond the scope of this paper, but it is fruitful to analyze methods that may be applied to construct new features from typical data.

3 Feature Construction

A set of raw initial features is provided as a part of problem description; out of these some may be directly useful and should be selected and tested. More features may be created by various transformations and pattern filters to extract useful information. A hierarchy of features with different complexity should be established, depending on the type of decision regions they provide and the complexity of search processes needed to discover them.

Binary features are the simplest and are a special case of projection on a line. Projections may be either unrestricted (all data are used for the projection), giving global features, or restricted, providing local features. Raw binary features $\{b_i\}$ should be regarded as global, resulting from unrestricted projections on a line. They are created from other features by dividing nominal features into two subsets, or creating subintervals of real features $\{x_i\}$ using decision trees or Quality of Projected Clusters [19] criteria.

Local binary features may be obtained by imposing various restrictions on projections. For binary features this will lead to complexes $b_1 \wedge b_2 ... \wedge b_k$ that help to distinguish interesting regions in feature space. For real-valued features restrictions based on intervals create hyperboxes $\prod_i [r_i^-, r_i^+]$. Restricted binary features are created by decision trees, for example if there is top level path $z = (x_1 < t_1) \wedge (x_2 \geq t_2)$, then z is a binary (logical) feature made from projection on x_2 restricted by $x_1 < t_1$ (or vice versa). Such features are especially useful for problems with inherent logical structure [7,8]. Logical features may be smoothed and changed into real, strongly non-linear feature using sigmoidal functions $z = \sigma(t_1 - x_1)\sigma(x_2 - t_2)$. Other ways to restrict subspace used for projection may be considered, for example taking only vectors that are not far from x_1 line and binarizing the projection $z = \sigma(x_1 - t_1)\sigma(d - ||\boldsymbol{x}||_{-1})$, where $||\boldsymbol{x}||_{-1}$ norm excludes feature x_1. More general version of binary features may use different restrictions for each values $b = 0$ and $b = 1$. Similar considerations may be done for nominal features.

The real-valued raw features are sometimes directly relevant to the learning task. Enhancement of local contrast is very important in natural perception. Features transformed by a steep sigmoidal function $\sigma(\beta x_i - t_i)$ are frequently more useful, closer to

binary. Slopes βx_i and thresholds t_i may be individually optimized using mutual information or other relevance measures independently for each feature. Single features may also show interesting patterns of $p(X|C)$ distributions, for example k relatively large groups of values, each corresponding to the same type of objects. Such k-separable solutions are very useful for learning complex Boolean functions. For example, n-bit parity functions are $n+1$-separable if a sum of all feature values is used as a new feature. A single cluster of pure cases is worth using as a new feature. Such features are generated by applying localized window-type functions to original features [19], for example $z_i = \sigma(x_i - a_i) - \sigma(x_i - b_i), a > b$.

Thus original real features may be pre-processed in several ways to increase their usefulness. The same is true for restricted, or partially localized real features. Instead of accepting raw features additional conditions restricting the subspace from which projections are made are added, $z_{ij} = x_i$ for $|x_j| < t_j$. Linearization of 2-dimensional distributions is possible using step-functions. If the minimal value of x_i is $x_{\min}$ and $z_{ij} = x_i \Theta(t_j - x_j) + (x_{\min} - 1)\Theta(x_j - t_j)$ all projections for $x_j \geq t_j$ are moved to $z_{ij} = x_{\min} - 1$. Consider for example fuzzy XOR relations between (x_i, x_j); such transformation will convert it to a 4-separable problem that is easily solved by most machine learning systems.

More complex line patterns and higher-dimensional patterns may be considered. First, linear combinations of features provide frequently better features than raw features. The simplest and computationally cheapest way to generate them is to start from calculation of the class centers $\boldsymbol{m}_i$, and use normalized projection directions $\boldsymbol{w}_{ij} = (\boldsymbol{m}_i - \boldsymbol{m}_j)/||\boldsymbol{m}_i - \boldsymbol{m}_j||$. If conditional probabilities $p(\boldsymbol{x}|C)$ have approximately Gaussian distributions $z(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{x}$ captures all information and other features will be spurious. This is indeed the case in a number of benchmark data, making them trivial. Drawing $p(C|z)$ shows areas of significant overlaps and allows for separation of border vectors that fall near the threshold $p(C_i|z) = p(C_j|z)$. Using only those vectors more projection directions may be generated; first class-dependent clusterization of these vectors is done and then search for pairs of cluster centers from different classes that are close to each other.

In recent years kernel-based learning methods dominate in pattern recognition [1]. Kernels are used to measure similarity to reference vectors providing new features $z_i = K(\boldsymbol{x}^{(i)}, \boldsymbol{x})$. Gaussian kernels are most popular, creating features that provide localized receptive fields, measuring how far vector $\boldsymbol{x}$ is from the reference support vector $\boldsymbol{x}^{(i)}$. Suppose that distribution $p(\boldsymbol{x}) = p(x_1, x_2)$ has been created as a sum of two partially overlapping Gaussian distributions; than transforming this distribution to (z_1, z_2), with kernels at the centers of clusters $z = z_1 + z_2$ variable will be almost constant along $p(\boldsymbol{x})$ =const, making the non-linear decision border almost flat after transformation. Quadratic optimization procedures in SVM may be replaced by any large-margin linear discrimination techniques, the extraction of useful information by kernels is primary reason for success.

Explicit generation of features based on different similarity measures [20] removes one SVM bottleneck, allowing for optimization of resolution in different areas of the feature space: strong non-linearities are provided where they are needed (small range of localized kernels), and using smooth functions (large range) when this is sufficient.

This technique may be called **adaptive regularization**, in contrast to simple regularization based on minimization of the norm of the weight vector $||\boldsymbol{w}||$ used in SVM or neural networks [1]. Although simple regularization enforces smooth decision borders decreasing model complexity it is not able to find the simplest solutions and may easily miss the fact that a single binary feature contains all information. Generation of kernel features should therefore proceed from most general, providing almost hyperplanar decision borders, with centers placed far from decision borders (identified looking at the $z = \boldsymbol{w} \cdot \boldsymbol{x}$ distribution for $\boldsymbol{w} = (\boldsymbol{m}_1 - \boldsymbol{m}_2)/||\boldsymbol{m}_1 - \boldsymbol{m}_2||$ direction), to more specific, highly non-linear, with non-zero contributions only close to decision border. More sophisticated features may also be useful: based on Mahalonobis distance calculated for clusters of vectors located near decision borders (an inexpensive method for rotation of density functions with d parameters has been introduced in [11]), or with flat local fronts using cosine distance.

Summarizing, the (incomplete) list of feature constructors includes:

- B1: Binary – unrestricted projections; provides 4 regions $p(C|b), C = 1, 2; b = 0, 1$, for MAP classifiers, complexity $O(1)N_b$.
- B2: Binary – restricted by other binary features; complexes $b = b_1 \wedge b_2 ... \wedge b_k$; also 4 regions $p(C|b)$, full complexity $O(2^k N_b)$, but only $O(N_b)$ with greedy algorithms.
- B3: Binary – restricted by distance; $b = 0 \wedge r_1 \in [r_1^-, r_1^+] ... \wedge r_k \in [r_k^-, r_k^+]$; search is done separately for each b value.
- N1: Nominal – like binary, with two or more subsets.
- R1: Line – original feature x_i, unrestricted projection $x_i(\boldsymbol{x})$; thresholds or intervals may change it into B1; contrast enhancements $\sigma(x_i)$, search for 1D patterns.
- R2: Line – like R1 but restricted by other features, $x_i = x_i(\boldsymbol{x})$ only for $|x_j| < t_j$.
- R3: Line – like R2 but restricted by distance, , $x_i = x_i(\boldsymbol{x})$ only for $\sum_{j \neq i} x_j^2 < t$.
- R4: Line – linear combination $z = \boldsymbol{w} \cdot \boldsymbol{x}$ optimized by unrestricted, projection pursuit (PCA, ICA, QPC ...); otherwise treated like R1.
- P: Prototype-based localized features $q(\boldsymbol{x}) = \exp(-||\boldsymbol{x} - \boldsymbol{r}||)$, weighted distance functions or specialized kernels.
- M: Motifs, based on correlations between elements (sequences, Hebbian correlations).
- T: Non-linear transformations: radial, separable, universal, rational, and other types of transfer functions [11] $f(\boldsymbol{x}) = \Phi(\boldsymbol{x}; \boldsymbol{w})$ for feature construction.

Prototype-based local features are created using support vectors near the decision border. First, projections lines $z_i = \boldsymbol{w}_i \cdot \boldsymbol{x}$ are generated using class means, and those vectors that are projected in the interval $z_i \in [\theta_i - r, \theta_i + r]$ selected, where $r = 0.05|\max z_i - min z_i|$, and the threshold $p(C|z_i = \theta_i) = p(C'|z_i)$. More line projections may be generated by grouping vectors into several clusters in each class and using restricted line projections for their means. Distances between selected vectors are evaluated (Euclidean distances are used) and only those with the fraction $[0.5 + \epsilon, 1 - \epsilon]$ of neighbors from the same class (with $\epsilon = 0.1$) in the σ radius are left. They are close to the decision borders, but selecting those with largest σ and clusterizing them to form features $q_i(\boldsymbol{x}) = \exp(-|\boldsymbol{x} - \boldsymbol{p}_i|/\sigma)$ will provide relatively smooth local features. Those vectors that are close to the border but have very small $q_i(\boldsymbol{x})$ are used again to create

features with smaller σ, localized more strongly, until most vectors are covered by local features. In this way multi-resolution kernel projection is realized with few support vectors. Complexity of this process is dominated by distance calculation, but this is done only on a small subset of all vectors, all other steps are linear in the number of vectors.

Second-order features may be build by learning machines using features that have been constructed and selected for learning. Ranking of local features should be done with care as they may not be globally important, but will be needed for local representation of information only. ULM may be presented as a constructive network, with new nodes representing transformations and procedures to extract useful features, and additional layers analyzing the image of data in the feature space created in this way. Here feature generators and classifiers are separated. ULM starts from original features, testing consecutively binary, nominal and real features, then testing the results of adding restricted features, and constructing new features using projections and prototypes. This process may be viewed as a search in the space of various feature spaces, analogous to the search in the space of similarity-based models [4,5].

4 Illustrative Results

The ULM approach is illustrated using a few benchmark datasets (taken from the UCI repository [21]) that create various problems to the state-of-the-art classification systems. Three quite different systems have been used in comparison: SVM with linear and Gaussian kernels (with proper choice of P-features giving identical results), Naive Bayes (NB) classifier, and Separability-Split Value (SSV) decision tree [22], all implemented in the Ghostminer software [23]. 10-fold stratified crossvalidation tests have been done in all cases.

Reference results obtained with original features are in Table 1. In parenthesis the number of generated features is noted. Line projections R1 were generated in two ways, from calculation of the class centers or cluster centers. For Hypothyroid, Wisconsin and Australian class centers worked better, for the remaining of datasets cluster centers were used. The simplest binary features B1 extracted from decision trees are presented in Table 2.

The *Australian credit* problem (15 features, 690 samples) has been used in the Statlog project [24] that compared 22 classifiers. The best result 86.9±, has been obtained with Cal5 tree (6 leaves on average), a rather fortuitous result as this tree has never been among top three for the remaining 21 datasets used in the Statlog study, the variance in crossvalidation test is about 4%, and the results have not been properly averaged over many runs. SVM with Gaussian kernel, optimized C=0.01 and $\sigma = 0.01$, using 460 support vectors, reaches 86.2±4.1%. Starting from ranking of single features it is easy to notice that A9 (our guess is that it codes "has bank account") is by far the most important feature (ex. Pearson correlation coefficient is 0.72, while the next best feature has 0.46). Using A9 binary feature SVM results with linear kernel is 85.5±3.5% with 182 SVs used and SSV tree and NB classifier give the same accuracy. For SVM and SSV this is only slightly higher than the reference value, but for NB it is a significant (over 5%) improvement. SSV and NB solutions are equivalent to the maximum a posteriori (MAP) rule: IF A9=T then class $+$, A9=F then class $-$.

Adding more features from the original set degrades this result, also adding line projections decreases accuracy, but adding to the binary feature one prototype-based feature near the decision border improved all results in statistically significant way to 86.8%, with only 3 leaves for SSV. While adding other features did not improved the result of SVM and NB, for SSV combination of the A9 with one R1-feature and one P-feature gave even better result, 87.0±2.50 with just 4 leaves. For this dataset ULM finds not only the simplest solution, but also using additional P-feature finds the most accurate solution known so far.

The *Ljubliana cancer* data contains 286 cases, of which 201 are no-recurrence-events (base rate 70.3%) and 85 are recurrence-events (29.7%). There are 9 attributes, with 2-13 different values each. SVM with Gaussian kernel gives 73.8±4.3%, linear SVM 71.0±4.7%, MLP network reaches similar accuracy to Gaussian SVM. The most important feature is Degree-malignant$\in \{1, 2, 3\}$, that for the value 3 has more than half recurrence events. Restricting this nominal feature (equivalent to selection of one path from two-level decision tree) using the number of involved nodes (ranging from 1 to 9) creates a binary feature: $B1$ = Degree-malignant = 3 $\wedge$ involved nodes> 1. The accuracy of the MAP rule based on this single feature, as well as the 3 classifiers in our study is 76.3±6.5%. Such simple and comprehensible solutions are easily missed if complex models (that in this case are less accurate) are used.

Appendicitis is another small dataset [7], containing only 106 cases, 88 cases with acute appendicitis and 18 cases with other problems, with 8 features (results of medical tests). Adding a single binary restricted feature B2 derived from SSV decision tree (see Table 2) increased the accuracy of all methods by 4-5%, to 91.5%, at the same time reducing complexity of classifiers. Other types of features did not improve this result.

The *Cleveland heart disease* dataset contains 303 cases, 164 corresponding to healthy (54.1%) and the rest to heart problems of various severity, described by 6 continuous and 4 nominal features. Adding new features did not lead to NB improvement in any significant way, but SVM benefited slightly from adding 3 line projections (achieving 83.5±3.6, with 98 SVs), and using 3 prototype-based features (84.2±6.1).

The *Pima Indian diabetes* dataset has only 8 attributes, 768 instances, 500 (65.1%) healthy and 268 (34.9%) diabetes cases. This dataset was used in the Statlog project [24], with the best 10-fold crossvalidation accuracy around 77.7% obtained by logistic discriminant analysis. Small improvement (1.3%) is gained in NB by adding binarized real feature (Table 2), and also by adding 3 line projections. Slight SVM improvement is noticed only with 3 line projections and 4 prototype-based features, giving 78.5±3.6%.

The *Wisconsin cancer* dataset contains 699 cases, with 458 benign (65.5%) and 241 (34.5%) malignant cases, described by 9 attributes with integer value in the range 1-10. Binarization of F2 feature improves the result in insignificantly way, but using a single line projection improves SSV results by 2.2% and NB results by 1.2%, while SVM is improved slightly only after 4 P-features are added, reaching 97.2±1.9% with 45 SVs.

The *Hypothyroid* dataset has 3772 cases for training and 3428 cases for testing, but to use consistent methodology the data has been merged and 10-fold stratified crossvalidation used. There are 22 attributes (15 binary, 6 continuous), and 3 classes: primary hypothyroid, compensated hypothyroid and normal (no hypothyroid), with unbalanced class distribution in the merged data, 166, 368 and 6666 vectors in the normal class.

Results of SVM and especially NB are quite poor (Table 1), however adding one binary restricted feature greatly improves them, from 94.1 to 99.5±0.4% with 80 SVs for SVM, and from 41.3 to 98.1±0.8% for NB. Adding more features does not improve these results further.

Table 1. Reference results with original features; ULM significantly improved results are described in the text

Dataset	Classifier		
	SVM (#SV)	SSV (#Leafs)	NB
Australian	84.9±5.6 (203)	84.9±3.9 (4)	80.3±3.8
Ljubliana cancer	72.0±5.1 (168)	74.7±5.0 (3)	72.2±7.4
Appendicitis	87.8±8.7 (31)	88.0±7.4 (4)	86.7±6.6
Heart	82.1±6.7 (101)	76.8±9.6 (6)	84.2±6.1
Diabetes	77.0±4.9 (361)	73.6±3.4 (4)	75.3±4.7
Wisconsin	96.6±1.6 (46)	95.2±1.5 (8)	96.0±1.5
Hypothyroid	94.1±0.6 (918)	99.7±0.5 (12)	41.3±8.3

Table 2. B1 features extracted from the decision tree

Dataset	B1 features	
Australian	$F8 < 0.5$	$F8 >= 0.5$ & $F9 >= 0.5$
Ljubliana cancer	Degree-malignant = 3	Degree-malignant = 3 $\wedge$ involved nodes> 1
Appendicitis	$F7 >= 7520.5$	$F7 < 7520.5$ & $F4 < 12$
Heart	$F13 < 4.5$ & $F12 < 0.5$	$F13 >= 4.5$ & $F3 >= 3.5$
Diabetes	$F2 < 123.5$	$F2 >= 143.5$
Wisconsin	$F2 < 2.5$	$F2 >= 4.5$
Hypothyroid	$F17 < 0.00605$	$F17 >= 0.00605$ & $F21 < 0.06472$

In case of the Wisconsin breast cancer data linear combination of features is clearly better than using individual features separately in decision trees or in Naive Bayes; in case of the hypothyroid data it is quite the opposite.

5 Discussion

Universal Learning Machines should be based on meta-learning schemes: searching for the best models in the space of selected data transformations. Meta-learning in the framework of similarity-based models [4,5] has been previously used with considerable success. Here a search in the space of various feature transformations is explored. Of course one may combine the two meta-learning approaches and in addition expand the space of all admissible transformations. General framework for such meta-learning schemes has recently been proposed in [16,17]. However, even the simplest scheme to construct-select features is worth exploring, generating new features that, followed by

forward selection or even a simple filter method to rank all features, may dramatically simplify and improve results.

While a lot of efforts have been devoted to feature selection procedures [25] much less effort has been spent on feature construction [26]. Yet functioning of biological neural networks gives numerous examples of transformations realized by neural columns that extract non-linear features from incoming signals, reducing noise in data and learning to estimate similarity of responses. Neural columns learn to react not only to intensity, but also to specific structures in the incoming stimuli, solving complex perceptual problems. As a step towards universal learning machines various ways of extracting information from input data have been systematically described, from binary features and their enhancements, through combinations of features derived from simple decision trees, combinations of features estimated from clusters, localized clusters after projections, to kernel-based features at different level of granularity, allowing for adaptive regularization. This of course does not exhaust all possibilities for construction of informative features. Various projection pursuit networks that reduce dimensionality, including PCA, ICA and other techniques [27], create interesting combination of original raw features according to such criteria as maximum variance or independence.

Feature constructors described here go beyond linear combinations. Systematic explorations of features of growing complexity allows for discovery of simple models that more sophisticated learning systems will miss. Some benchmark problems have been found rather trivial, and have been solved with a single binary feature, one constrained nominal feature, or one new feature constructed as a projection on a line connecting means of two classes. Kernel-based features offer an attractive alternative to current kernel-based SVM approaches, offering multiresolution and adaptive regularization possibilities. Analysis of images, multimedia streams or biosequences will require even more sophisticated ways of constructing features based on simpler features. Although larger datasets should be analyzed it is quite clear that constructing new features in this way opens new possibilities to create simple, comprehensible and accurate data models.

References

1. Schölkopf, B., Smola, A.: Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
2. Duch, W., Maszczyk, T.: Almost random projection machine. In: Alippi, C., et al. (eds.) ICANN 2009, Part I. LNCS, vol. 5768, pp. 789–798. Springer, Heidelberg (2009)
3. Duch, W.: k-separability. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 188–197. Springer, Heidelberg (2006)
4. Duch, W., Grudziński, K.: Meta-learning: searching in the model space. In: Proceedings of the International Conference on Neural Information Processing, Shanghai, pp. 235–240 (2001)
5. Duch, W., Grudziński, K.: Meta-learning via search combined with parameter optimization. In: Rutkowski, L., Kacprzyk, J. (eds.) Advances in Soft Computing, pp. 13–22. Springer-Physica Verlag, New York (2002)
6. Duda, R.O., Hart, P.E., Stork, D.: Patter Classification. J. Wiley & Sons, New York (2001)
7. Duch, W., Adamczak, R., Grąbczewski, K.: A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks 12, 277–306 (2001)

8. Duch, W., Setiono, R., Zurada, J.: Computational intelligence methods for understanding of data. Proceedings of the IEEE 92, 771–805 (2004)
9. Grochowski, M., Duch, W.: Learning highly non-separable Boolean functions using Constructive Feedforward Neural Network. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 180–189. Springer, Heidelberg (2007)
10. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. In: Cognitive Technologies. Springer, Heidelberg (2009)
11. Duch, W., Jankowski, N.: Survey of neural transfer functions. Neural Computing Surveys 2, 163–213 (1999)
12. Duch, W., Jankowski, N.: Taxonomy of neural transfer functions. In: International Joint Conference on Neural Networks, Como, Italy, vol. III, pp. 477–484. IEEE Press, Los Alamitos (2000)
13. Duch, W.: Grąbczewski, K.: Heterogeneous adaptive systems. In: IEEE World Congress on Computational Intelligence, pp. 524–529. IEEE Press, Honolulu (2002)
14. Duch, W.: Towards comprehensive foundations of computational intelligence. In: Duch, W., Mandziuk, J. (eds.) Challenges for Computational Intelligence, vol. 63, pp. 261–316. Springer, Heidelberg (2007)
15. 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, pp. 33–36 (2003)
16. Grabczewski, K., Jankowski, N.: Versatile and efficient meta-learning architecture: Knowledge representation and management in computational intelligence. In: IEEE Symposium on Computational Intelligence in Data Mining, pp. 51–58. IEEE Press, Los Alamitos (2007)
17. Grabczewski, K., Jankowski, N.: Meta-learning with machine generators and complexity controlled exploration. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 545–555. Springer, Heidelberg (2008)
18. Duch, W.: Filter methods. In: Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. (eds.) Feature extraction, foundations and applications, pp. 89–118. Springer-Physica Verlag, New York (2006)
19. Grochowski, M., Duch, W.: Projection Pursuit Constructive Neural Networks Based on Quality of Projected Clusters. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part II. LNCS, vol. 5164, pp. 754–762. Springer, Heidelberg (2008)
20. Duch, W., Adamczak, R., Diercksen, G.: Classification, association and pattern completion using neural similarity based methods. Applied Mathematics and Computer Science 10, 101–120 (2000)
21. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
22. 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, pp. 201–208 (2000)
23. Duch, W., Jankowski, N., Grąbczewski, K., Naud, A., Adamczak, R.: Ghostminer data mining software. Technical report, Department of Informatics, Nicolaus Copernicus University (2000-2008), `http://www.fqspl.com.pl/ghostminer/`
24. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: Machine learning, neural and statistical classification. Elis Horwood, London (1994)
25. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.: Feature extraction, foundations and applications. Springer-Physica Verlag, Heidelberg (2006)
26. Liu, H., Motoda, H.: Feature extraction, construction and selection: a data mining perspective. In: Liu, H., Motoda, H. (eds.) SECS, vol. 453. Kluwer Academic, Boston (1998) (Includes bibliographical references and index)
27. Pękalska, E., Duin, R.: The dissimilarity representation for pattern recognition: foundations and applications. World Scientific, Singapore (2005)