Prototype rules from SVM

Marcin Blachnik[1] and Włodzisław Duch[2]

[1] Division of Computer Methods, Department of Electrotechnology, The Silesian University of Technology, ul. Krasińskiego 8, 40-019 Katowice, Poland; Email: Marcin.Blachnik@polsl.pl
[2] Department of Informatics, Nicolaus Copernicus University, Grudziądzka 5, Toruń, Poland; Email Google: Duch

1 Why prototype-based rules?

Propositional logical rules may not be the best way to understand the class structure of data describing some objects or states of nature. The best explanation may differ depending on the problem, the type of questions and the type of explanations that are commonly accepted in a given field. Although most research has focused on propositional logical rules [14, 19] their expressive powers have serious limitations. For example, a simple majority voting can be expressed using the "majority is for it" concept that is easy to formulate using M-of-N threshold rules. Given n binary $x_i = 0, 1$ answers the rule $\sum_{i=1}^{n} x_i > 0.5n$ is an elegant expression of such concept and is impossible to state directly in propositional form, leading to $\binom{n}{n/2}$ terms. This type of rules may be regarded as a particular form of similarity or prototype-based rules. In the voting example the similarity to the "all for it" prototype $\mathbf{A}$, that is a vector with all $a_i = 1$, has to be greater than $n/2$ in the Hamming distance sense, $||\mathbf{A} - \mathbf{X}|| < n/2$. Cognitive psychology experiments proved that human categorization of natural objects and states of nature is based on memorization of numerous examples and creation of prototypes that are abstractions of these examples [34]. Propositional logical rules are prevalent in abstract sciences but in real life they are rarely useful, their use being restricted to enumeration of small number of nominal values, or one or two continuous features with corresponding thresholds. In real life "intuitive understanding" is used more often, reflecting experience, i.e. memorized examples of patterns combined with various similarity measures that allow for their comparison and evaluation.

Decision borders between different categories produced by propositional rules are simple hyperboxes. Univariate decision trees provide even simpler borders based on hierarchical reduction of decision regions to half-spaces and hyperboxes. Using similarity to prototypes quite complex decision regions may be created, including hyperboxes and fuzzy decision regions. Some of these decisions may be difficult to describe using linguistic statements and thus may server as a model of intuition. One may argue that comprehensibility of rules is lost in this way, but if similarity func-

tions are sufficiently simple interpretation may in fact be quite easy. For example, interpretation of the $||\mathbf{A} - \mathbf{X}|| < n/2$ rule is quite obvious. Other voting rules may easily be expressed in the same way, including polarization of opinions around several different issues. Weighting evidence before decision is made requires non-trivial aggregation function to combine all available evidence, and similarity or dissimilarity functions are the most natural way to do it. Despite these arguments the study of prototype-based rules has been much less popular than of the other forms of rules.

Similarity-Based Methods (SBM) [8, 13] are quite popular in pattern recognition and data mining. The framework for construction of such methods enables integration of many methods for data analysis, including neural networks [12], probabilistic and fuzzy methods [15], kernel approaches and many other methods [32]. One of the most exciting possibility that such framework offers is to build the simplest accurate method on demand, in a meta-learning scheme, searching for the best model in the space of all similarity-based methods [18]. This family of methods includes also prototype-based rules (P-rules) [17] that are more general than fuzzy rules (F-rules), crisp propositional rules (C-rules) and M-of-N rules, including them as special cases. All methods covered by the SBM framework represent knowledge as a set of prototypes or reference vectors, adding appropriate similarity metrics and the aggregation procedures that combine information from different prototypes giving the final output. Several similarity-based transformations may be done in succession, creating higher-order SBM models. Prototype based rules are based on the SBM framework, but their aim is to represent the knowledge hidden in the data in the most comprehensible way. This goal is obtained by reducing the number of prototype vectors (prototype selection), minimizing the number of features used to create final model and using simple similarity metrics.

One of the most important advantages of P-rules is their universality. They enable integration of different type of rules, depending on the similarity function associated with each prototype: classical crisp rules result form Chebychev distance, fuzzy rules (F-rules) from any separable similarity metrics [16]. P-rules can also represent M-of-N rules in a natural way using prototype threshold rules [21, 2], adding the distance to a prototype as one of the coordinates. Such rules often give very simple interpretation of data, for example a single prototype threshold rule gives over 97.5% accuracy on a well known Wisconsin Breast Cancer dataset [21]. Thus P-rules provide most general form of knowledge representation.

Two general types of P-rules are possible, the Nearest Neighbor Rules (PN-rules), and the prototype threshold rules (PT-rules), introduced in the next section. In the third section the use of support vectors as prototypes is discussed. Reduction of the number of support vectors (SVs) and methods of searching for informative prototypes are described in section 4 and 5, while numerical examples are presented in section 6. Perspectives on the use of support vector machines for P-rule extraction conclude this paper.

2 P-rules and their interpretation

Prototype rules are based on analysis of similarity between objects and prototypes that are used as a reference. In its most general form [13, 32] objects (cases) $\{\mathbf{O^i}\}$, $i = 1..n$ do not need to be represented by numerical features, a kernel (or a set of different kernels that provide "receptive fields" that stress different perspectives) estimating (dis)similarity is sufficient $K_{ij} = K(\mathbf{O^i}, \mathbf{O^j})$ to characterize such objects. Selecting some of these objects as prototypes an object $\mathbf{O}$ is represented by n-dimensional vector $\mathbf{p}(\mathbf{O}) = \mathbf{K}^p$. Alternatively, each object is represented by N feature values. In the first case features come from evaluation of similarity and may be created for quite complex and diverse objects (such as proteins or whole organisms), for which a common set of features is hard to define. Below it is assumed that all objects are described by vectors in some feature space.

A single prototype $\mathbf{p}$ with associated similarity function $S(\cdot, \mathbf{p})$ defines for a given threshold θ a subspace $\mathcal{S}_p$ of vectors $\mathbf{x}$ for which $S(\mathbf{x}, \mathbf{p}) < \theta$. This subspace is centered at the position of the prototype $\mathbf{p}$ and may have different shapes, depending on the similarity function. Such interpretation defines a crisp logical rule for the new feature $x_p = S(\mathbf{x}, \mathbf{p})$. In this case the antecedent part of a P-rule uses similarity to a single prototype and the class label of that prototype (in classification tasks) is the consequence part.

$$\text{If } S(\mathbf{x}; \mathbf{p}) > \theta \text{ Then } C(\mathbf{x}) = C(\mathbf{p}) \tag{1}$$

The similarity value may be used to estimate confidence factor for such rule. The rescaled difference $\mu_p(\mathbf{x}) = S(\mathbf{x}, \mathbf{p}) - \theta$ may obviously be interpreted as a fuzzy membership function defining the degree to which vector $\mathbf{x}$ belongs to the fuzzy subspace $\mathcal{S}_p$. Many similarity functions are separable in respect to all features:

$$S(\mathbf{x}; \mathbf{p}, \sigma) = \prod_i S(x_i, p_i; \sigma_i) \tag{2}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and $\mathbf{p} = [p_1, p_2, \ldots, p_n]^T$ are n-dimensional vectors, and $S(\cdot)$ is similarity function.

Threshold P-rules with separable similarity functions can be interpreted as fuzzy rules (F-rules) with a product as a fuzzy and aggregation operator. Linguistic interpretation of F-rules relies on semantics of linguistic values assigned to each linguistic variable as adjectives describing the membership functions. Such representation is sensitive to context. Good example of this context dependence is an adjective *high* that may describe objects of different types, for example *a person*, but even in this case different kinds of people: kids, women or basketball players will require different membership function representing variable "high". Thus indirectly fuzzy rules have to rely on prototypes of objects or concepts to define the context, but since in fuzzy rules this context is not explicitly represented confusion is quite likely. P-rules make this reliance explicit always pointing to prototypes of particular concepts, allowing each concept to be decomposed into independent features that may be treated as linguistic values in the fuzzy sense.

2.1 Types of P-rules

Two distinct types of P-rules are:

- Prototype Threshold Rules (PT-rules), where each prototype $\mathbf{p}_i$ has an associated threshold θ_i value and i-th rule is written as:

$$\text{If } S(\mathbf{x}, \mathbf{p}_i) > \theta_i \text{ Then } C(\mathbf{x}) = C(\mathbf{p}_i) \tag{3}$$

  where $C(\cdot)$ is a function returning class labels or some other information associated with the prototype.
- Nearest Neighbor Rule (PN-rules), where the most similar prototype is selected:

$$\text{If } k = \arg\max_i S(\mathbf{x}, \mathbf{p}_i) \text{ Then } C(\mathbf{x}) = C(\mathbf{p}_k) \tag{4}$$

  so the output value depends on the internal relations between prototypes.

More general form of PN-rules is used by the Generalized Nearest Prototype Classifier [25]. From the rule-based perspective it is defined as: If $\mathbf{x}$ is similar to $\mathbf{p}_i$ then it is of the same class with some support w_i:

$$\text{If } w_i = S(\mathbf{x}, \mathbf{p}_i) \text{ Then } C(\mathbf{x}) = C(\mathbf{p}_i) \text{ with support } w_i \tag{5}$$

where $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_v]^T$ is set of v prototype vectors, and w_i is support for the conclusion of the i-th rule. The final decision of the set of such rules is obtained as:

$$C(\mathbf{x}) = A(w_i, C(\mathbf{p}_i)) \tag{6}$$

where $A(\cdot)$ is an aggregation operator, which joins conclusions of individual P-rules.

2.2 Support vectors as prototypes

The SVM model defines a hyperplane for linear discrimination in the feature space:

$$\mathbf{\Psi} = \sum_{i=1}^{m} \gamma_i \phi(\mathbf{x}_i) \tag{7}$$

where $\phi(\mathbf{x})$ is function that maps vectors from n dimensional input space to some feature space F. Since scalar products are sufficient to define linear models in the ϕ-transformed space kernels are used to represent these products in the original feature space. Decision function is in this case defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \gamma_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{8}$$

where m is the number of support vectors $\mathbf{x}_i$ with non-zero γ_i coefficients (Lagrangian multipliers), $K(\mathbf{x}, \mathbf{x}_i)$ is the kernel function, and $y_i = C(\mathbf{x_i}) = \pm 1$ are the class labels.

This model may be expressed as a set of PN-rules with weighted aggregation $A(\cdot)$ (Eq. (5)) as a sum from $i = 1$ to m, replacing the kernel with a similarity function $S(\cdot, \cdot)$ and defining support for a rule as $w_i = \alpha_i S(\mathbf{x}, \mathbf{p_i}; \alpha)$. Similar ideas have also been considered from the fuzzy perspective by Chen and Wang [5] who interpret SVM model as a fuzzy rule based system. In their paper they introduced Positive-Definite Fuzzy Classifiers using the Takagi Sugeno (TS) fuzzy inference system [37], adopting this model to extract fuzzy rules from support vector machines. However, in their solution comprehensibility and model transparency, the most important properties of any rule bases system, are lost. As stated in [19], logical rules are useful only if they are simple and accurate, otherwise there is no point in extracting rules from black box systems that works well because no additional understanding is gained by creation of many complex rules. The goal of comprehensibility and transparency can be achieved only when small number of support vectors (SV) can be defined, or when SVM decisions can be replicated with another simpler rule-based model. These two strategies have been studied by many research groups. The first leads to methods aimed at reduction of the number of support vectors through removing approximately linearly dependent kernels in the SV set. The second one leads to the "Reduced Set" methods aimed at reconstruction of the SVM hyperplane defined in the kernel feature space with smaller number of kernel functions.

The initial idea behind the kernel reduction methods was to speed up the decision process, but these methods can obviously be also used to understand data using small number of P-rules. These two approaches differ in the way support vectors are used. SVs are vectors that define or lie within the margin, that is they are close to the decision hyperplane. Those SVs that are outside of the margin on the wrong side should be removed, as they are cases that cannot be correctly classified and should not be used to create rules. Reducing linear dependencies removes some of the original SVs but will leave other SVs intact. In the "Reduced Set" approach new support vectors are not selected from the training examples but may be defined anywhere in the input space.

2.3 Removing linear dependencies among support vectors

Many numerical methods of removing linear dependencies from the kernel matrix $K(\mathbf{x}_i, \mathbf{x}_j)$ may be defined. For smooth kernels the problem may also be analyzed in the feature space, because it is created by vectors that are too similar to each other. Therefore clusterization techniques may be used to select representative vectors that are sufficiently distinct to avoid problems with linear dependencies. In some applications cluster centers may replace original vectors.

SVM approach based on quadratic programming has a unique solution, avoiding the problem of local minima and initialization of parameters that neural network algorithms have to face. Still there are some differences in SVM implementations that use different quadratic programming solvers. Solutions obtained with SMO [33], SVM Light [24], SVMTorch [6] or other SVM methods slightly differ from each other. On the other hand even if identical decision function are obtained different support vectors may be selected during the optimization procedure. Such situation

is bound to happen when SVs are linearly dependent. This observation leads to a reduction of the number of support vectors, as studied by Downs et al. in [7] in the algorithm referred below using RLSV acronym (removed linearly-dependent support vectors). Linear dependence in the kernel space Φ can be written as:

$$\phi(\mathbf{x}_k) = \sum_{\substack{i=1 \\ i \neq k}}^{v_x} q_i \phi(\mathbf{x}_i) \tag{9}$$

where q_i are scalar coefficients. If such vector $\mathbf{x}_k$ exist (up to predefined precision) the decision hyperplane (7) can be rewritten as:

$$\mathbf{\Psi} = \sum_{\substack{i=1 \\ i \neq k}}^{m} \gamma_i y_i \phi(\mathbf{x}_i) + \gamma_k y_k \sum_{\substack{i=1 \\ i \neq k}}^{m} q_i \phi(\mathbf{x}_i) + b \tag{10}$$

Using the kernel equation (10) is written as:

$$f(\mathbf{x}) = \sum_{\substack{i=1 \\ i \neq k}}^{m} \gamma_i y_i K(\mathbf{x}, \mathbf{x}_i) + \gamma_k y_k \sum_{\substack{i=1 \\ i \neq k}}^{m} q_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{11}$$

This may be finally rewritten as:

$$f(\mathbf{x}) = \sum_{\substack{i=1 \\ i \neq k}}^{m} \gamma_i' y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{12}$$

where

$$\gamma_i' = \gamma_i + \gamma_k q_i y_k / y_i \tag{13}$$

The form of the decision function is thus unchanged, but the coefficients are redefined to account for the removed component.

2.4 Reducing the number of support vectors

The methodology of reduced set methods was proposed by Burges in [4]. When support vectors are removed the dimensionality of the transformed space is decreased and this is reflected in the change of the original decision hyperplane $\mathbf{\Psi}$ in the input space to $\mathbf{\Psi}'$ plane. The distance between the two hyperplanes

$$d = \min ||\mathbf{\Psi} - \mathbf{\Psi}'||^2 \tag{14}$$

should be as small as possible, and the approximation $\mathbf{\Psi}'$

$$\mathbf{\Psi}' = \sum_{i=1}^{m'} \beta_i \phi(\mathbf{x}_i) \tag{15}$$

for the P-rules should satisfy $m' \ll m$, with scalar coefficients β_i. The inequality $m' \ll m$ should be considered very carefully because the number of SV cannot be too small [27].

Now there are two possible solution to the problem stated in this way. First, both the coefficients β_i and the position of vectors $\mathbf{x}_i$ in the input space may be optimized, and second, only the coefficients are optimized while support vectors are kept selected from the input vectors $\mathbf{x}_i$. Both approaches has some advantages and disadvantages. Optimization of SV positions allows for better approximation and thus stronger reduction of the number of support vectors, but may lead to vectors that are difficult to interpret from the P-rule perspective. For example, in medical applications unrestricted optimization of support vector positions may create cases that are quite different from real patient's data, including intermediate values of binary features (such as sex). A compromise in which optimization of SVs is performed only in selected dimensions may be the best solution from both accuracy and comprehensibility point of view.

Optimizing support vectors $\mathbf{z}_i$ requires minimization of (14) over β and $\mathbf{z}$:

$$\begin{aligned} \min_{\beta,\mathbf{z}}(d) = \textstyle\sum_{i,j=1}^{m} \gamma_i \gamma_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i,j=1}^{m'} \beta_i \beta_j K(\mathbf{z}_i, \mathbf{z}_j) \\ -2 \sum_{i=1}^{m} \sum_{j=1}^{m'} \beta_j \gamma_i K(\mathbf{x}_i, \mathbf{z}_j) \end{aligned} \tag{16}$$

Directly minimization [4] requires evaluation of derivatives:

$$\frac{\partial}{\partial \beta_a} \left\| \mathbf{\Psi} - \sum_{i=1}^{m'} \beta_i \phi(\mathbf{z}_i) \right\|^2 = 2\phi(\mathbf{z}_a) \left(\mathbf{\Psi} - \sum_{i=1}^{m'} \beta_i \phi(\mathbf{z}_i) \right) \tag{17}$$

Setting this derivative to zero and replacing $\mathbf{\Psi}$ with (7) one obtains:

$$\sum_{j=1}^{m} \gamma_j \phi(\mathbf{x}_j) = \sum_{i=1}^{m'} \beta_i \phi(\mathbf{z}_i) \tag{18}$$

In the kernel matrix notation $K^{zx}\boldsymbol{\gamma} = K^{zz}\boldsymbol{\beta}$ where $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_m]^T$, $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_{m'}]^T$, and K^{zx} is matrix of the $m' \times m$ dimensions containing $K(\mathbf{z}_i, \mathbf{x}_j)$ values. The solution may be written in a number of ways, for example:

$$\boldsymbol{\beta} = \left(K^{zz}\right)^{-1} K^{zx} \boldsymbol{\gamma} \tag{19}$$

or using psuedoinverse matrices etc. Selection of the support vectors $\mathbf{z}_i$ from the initial pool of SVM-selected input vectors can be done using systematic search techniques, or using some stochastic selection procedures.

An interesting procedure for approximation $\mathbf{\Psi}$ have been proposed in [35], where the problem has been analyzed as clustering in the feature space. First notice that instead of direct distance (17) minimization the distance between $\mathbf{\Psi}$ and orthogonal projection of $\mathbf{\Psi}$ on the space generated by $Span(\phi(\mathbf{z}))$ may be used. Considering a single vector $\mathbf{z}$ and equation (16) the value of β is calculated from (19) as:

$$\beta = \sum_{i=1}^{m} \gamma_i K(\mathbf{x}_i, \mathbf{z}) \Big/ K(\mathbf{z}, \mathbf{z}) \tag{20}$$

and then $\mathbf{z}$ may be optimized minimizing:

$$\min_{\mathbf{z}} \left\| \frac{\mathbf{\Psi} \cdot \phi(\mathbf{z})}{\phi(z)\phi(\mathbf{z})} \phi(\mathbf{z}) - \mathbf{\Psi} \right\|^2 = \|\mathbf{\Psi}\|^2 - \frac{\left(\mathbf{\Psi} \cdot \phi(\mathbf{z})\right)^2}{\phi(\mathbf{z})\phi(\mathbf{z})} \tag{21}$$

This is equivalent to maximization of:

$$\max_{\mathbf{z}} \left(\frac{\left(\mathbf{\Psi} \cdot \phi(\mathbf{z})\right)^2}{\phi(\mathbf{z})\phi(\mathbf{z})} \right) \tag{22}$$

In case of similarity-based kernels $K(\mathbf{z}, \mathbf{z}) = 1$ and maximization in (22) can be simplified just to maximization of the numerator using fixed-point iterative methods. Calculating derivatives it is not hard to show that first approximation to $\mathbf{z}_1$ is calculated as [35]:

$$\mathbf{z}_1 = \frac{\sum_{i=1}^{m} \gamma_i K(||\mathbf{x}_i - \mathbf{z}||^2)\mathbf{x}_i}{\sum_{i=1}^{m} \gamma_i K(||\mathbf{x}_i - \mathbf{z}||^2)} \tag{23}$$

and iterations improve this estimation:

$$\mathbf{z}_{n+1} = \frac{\sum_{i=1}^{m} \gamma_i K(||\mathbf{x}_i - \mathbf{z}_n||^2)\mathbf{x}_i}{\sum_{i=1}^{m} \gamma_i K(||\mathbf{x}_i - \mathbf{z}_n||^2)} \tag{24}$$

Stability of this process is not guaranteed, and results strongly depend on the initialization of $\mathbf{z}$ and may require multiple restarts to find good solution. This is one of many possible approaches. Another interesting method has been proposed by Kwok and Tsang [26], using Multidimensional Scaling (MDS) algorithm to represent images of the feature space vectors back in the input space.

2.5 Finding optimal number of support vectors

Analysis of numerical experiments performed by Downs et al. [7] shows that RLSV method is not sufficient for rule generation. Elimination of linear dependencies

among SVs leads to a small reduction of their number, although quality of results is usually quite good. One exception is reduction of over 80% of the original number of SVs for the Heberman dataset [7], where quadratic kernel with SMO optimization found 87 SVs, while RLSV algorithm reduced it to just 10 vectors. Stronger reduction may be achieved relaxing numerical accuracy for linear dependency tests, but this will probably degrade also the quality of results. The effects of such reduction remains to be investigated.

Quality of this method depends on the type of kernel function, the C-value and the complexity of the decision border created by the SVM algorithm. Parameter C defining the size of SVM margins has important influence on the number of SVs. In soft margin SVM harder margins (lower C value) leads to a higher number of SVs that have more linear dependencies and thus higher reduction rate is obtained. Generally best results of RLSV algorithm are obtained for linear kernel, as in principle two support vectors are sufficient to define a decision hyperplane.

RS-SVM approach allows for significant reduction of the number of SVs, leading to more comprehensible models. To find optimal number of SVs any search method can be used with typical cost function driven by minimization of the distance between separating hyperplanes ((14)):

$$E_1(m') = \|\mathbf{\Psi} - \mathbf{\Psi}'\| = \left(\sum_{i,j=1}^{m} \gamma_i \gamma_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i,j=1}^{m'} \beta_i \beta_j K(\mathbf{z}_i, \mathbf{z}_j) - 2 \sum_{i=1}^{m} \sum_{j=1}^{m'} \beta_j \gamma_i K(\mathbf{x}_i, \mathbf{z}_j) \right)^2 \tag{25}$$

An additional term $\alpha m'/m$ defining model complexity as a fraction of reduced number of SVs (m') to the original number of SVs (m) multiplied by some constant α may be added to the difference of distances between hyperplanes. Because the distance $\|\mathbf{\Psi} - \mathbf{\Psi}'\|$ may take very high values α may be rescaled by $1/\|\mathbf{\Psi} - \mathbf{\Psi}'_1\|$, where $\mathbf{\Psi}'_1$ is $\mathbf{\Psi}'$ defined with just one SV.

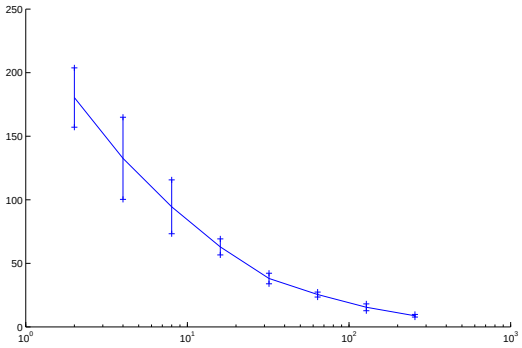
An alternative function that measures changes in classification accuracy may be defined as:

$$E_2(m') = \text{acc(SVM)} - \text{acc(RSSVM(m'))} \tag{26}$$

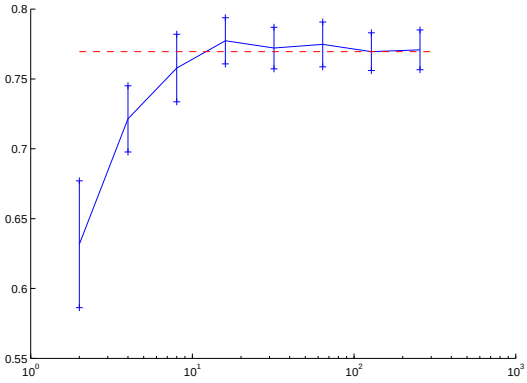
where acc() is classification accuracy measured using some loss function; in this case also the penalty for complexity may be added. Because $acc(SVM)$ doesn't change during optimization the number of SV, we can simplify the function (26) omitting the first component, optimizing:

$$E_3(m') = \text{acc(RSSVM(m'))} \tag{27}$$

To compare the approach based on minimization of distance and accuracy few tests have been done using Gaussian SVM on two datasets, Pima Indians Diabetes, and Cleveland Heart Disease [28]. In the first step all datasets were normalized to the $[0, 1]$ range. The best C value for the SVM was selected using 5-fold cross validation (CV) greedy search procedure in the $C = 2^1$ to $C = 2^8$ range, while $\sigma = 1$, and



(a) Dependence of the cost function E_1 on the logarithm of the number of SVs

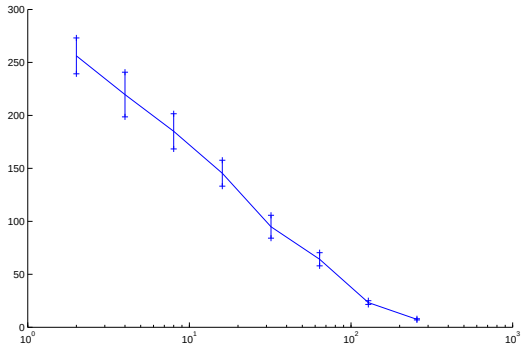


(b) Dependence of the mean accuracy (cost function E_3) on the logarithm of the number of SVs. Dashed line represents mean accuracy of the original SVM

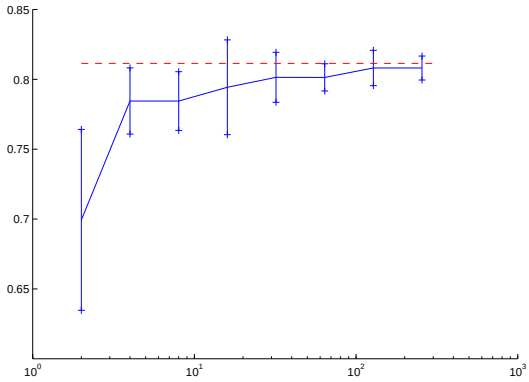
Fig. 1. Comparison of the distance (Eq. (26)) and accuracy (Eq. (27)) based cost functions for Pima Indians diabetes data.

Alpha cutoff=10^{-2} were fixed. Finally the process of 5-fold CV was used to test different cost functions using Fixed Point Iteration algorithm (Fig. (1) for the first, and Fig. (2) for the second dataset). The distance between hyperplanes plotted in the top subfigure is decreasing in approximately linear way with the logarithm of

the number of SVs. On the other hand the classification accuracy (Eq. (27)) grows rapidly reaching the accuracy of SVM with just a few SVs.



(a) Dependence of the cost function E_1 on the logarithm of the number of SVs



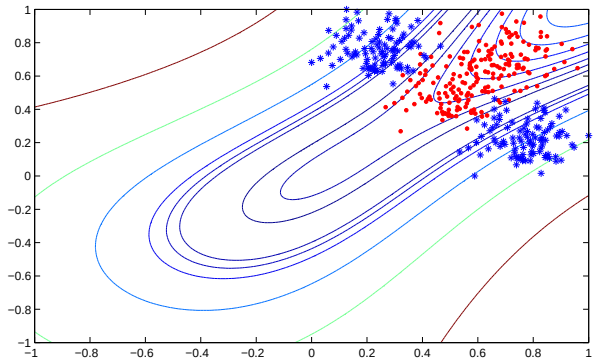
(b) Dependence of the mean accuracy (cost function E_3) on the logarithm of the number of SVs. Dashed line represents mean accuracy of the original SVM

Fig. 2. Comparison of the distance (Eq. (26)) and accuracy (Eq. (27)) based cost functions for Cleveland Heart disease data.

Although increasing the number of SVs leads to decision border that are equivalent to the one found by SVM algorithm without restrictions on the number of SVs results are not correlated with increasing accuracy of the models. Large differences between hyperplanes in the region far from data are not important, but the distance-based approach does not distinguish between different regions, trying to decrease the overall distance. This problem will be especially acute for Gaussian or other non-linear kernels that place SV far from decision borders in the feature space. For two overlapping distributions SVM with Gaussian kernels will use support vectors that are all around both distributions, even though only those that are close to the support vectors from the opposite class are really useful. It should be possible to use the distance between closest support vectors from the opposite classes to rank candidates for removal in the SV selection process. This can simplify the search in the accuracy-based approach.

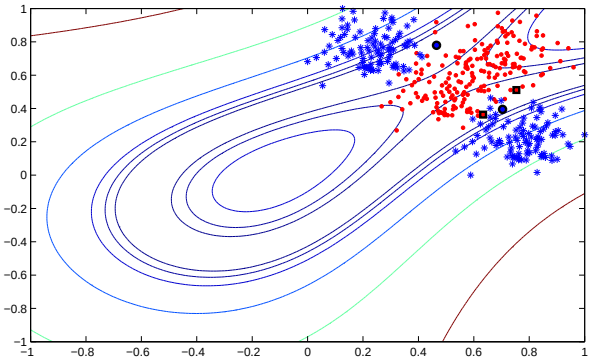
2.6 Problems with interpretation

Even if a simple and transparent model that mimics SVM's decision borders could be created the question "what can be learned from it" still remains. Similar problems face most rule extraction approaches, including fuzzy and rough rule based systems, with the exception of simple crisp rule sets that sometimes have straightforward interpretation [14, 19]. Prototype-based rules demand not only a small number of prototypes but also a meaningful position of these prototypes among other input vectors.

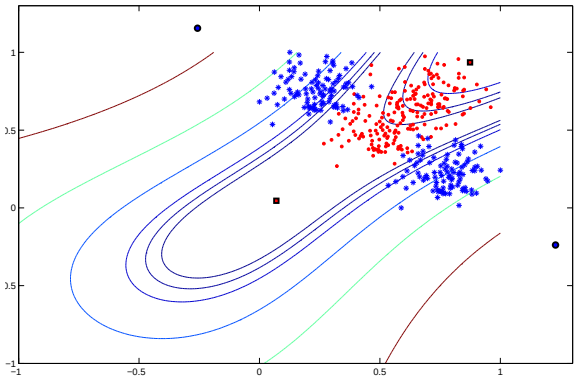


(a) Contour plot of SVM classifier decision borders

None of the support vector reduction methods considered here gives prototypes which have simple interpretation, as they are never placed at the centers of clusters



(b) Contour plot of Schölkopf's RS-SVM with marked positions of prototypes



(c) Contour plot of Burges RS-SVM with marked positions of prototypes

Fig. 3. An example of decision borders generated by a) SVM classifier, b) with RS-SVM reduction according to Schölkopf and c) Burges algorithm.

(as in the RBF networks). This problem is illustrated in figure (3(a)). Four prototypes selected by the Schölkopf algorithm are somewhere near the decision border and in the "flattened" image space are sufficient to define good border, but in the feature spaces they make little sense. More intuitive solution is obtained with the Burges

algorithm where position of prototypes looks more "natural", however also here the knowledge which can be inferred from these positions is not clear.

If the goal is to understand the data the problem of prototype selections should be solved in some other way. In the next section algorithms driven by prototype selection methods used in the k-nearest neighbor (kNN) classifiers are used to search for informative prototypes.

3 Searching for informative prototypes

SVM decision borders should be approximated in such a way that uses informative prototypes to understand data structure. These prototypes do not have to be selected from support vectors, but may be placed in optimized positions. Possible solutions can be taken from kNN learning algorithms where many prototype selection methods that reduce the number of reference vectors exist. Good comparison of existing prototype selection algorithms can be found in papers by Jankowski and Grochowski [23, 22] and Wilson and Martinez [39]. The general algorithm proposed here starts from training SVM model, then selecting prototypes using one of the algorithms developed for kNN methods, and then assigning to each prototype weight value to reproduce the SVM decision border. The weights are calculated using equation (19). To facilitate better interpretation of results weights can be normalized without any loss of generalization using softmax procedure:

$$\boldsymbol{\beta}' = \frac{\boldsymbol{\beta}}{\sum \boldsymbol{\beta}} \tag{28}$$

The weight value after normalization indicates how strong is the influence of each prototype on the final decision function. Generally the higher β_i' is, the more important associated i'th prototype is. The algorithm is schematically written below.

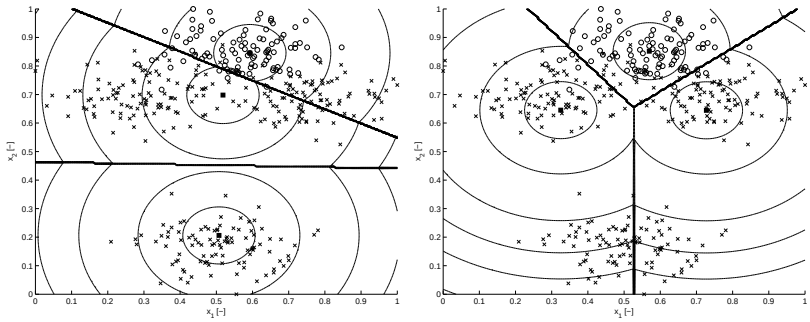
Algorithm 1

1: train SVM;
2: select prototypes with one of the kNN-based algorithms;
3: optimize prototype weights using formula (19);
4: normalize weights to [0,1] range.

3.1 Prototype selection using context dependent clustering

One of the most popular methods for prototype selection in kNN and RBF classifiers is to use clustering of the training vectors. However, unsupervised clustering algorithms do not use any knowledge about class structure, leading to unnecessarily large number of prototypes. Such situation is presented in figure (4), where one of the prototypes is useless because it does not participate directly in construction

of the decision border. This problem may be solved with semi-supervised clustering. A clustering approach which uses additional knowledge to reduce the number



(a) Prototype selection using classical clustering method (FCM)

(b) Prototype selection using context clustering (CFCM)

Fig. 4. Comparison of prototype selection methods using two types of clustering methods, FCM and CFCM

of prototypes was proposed by Blachnik et al. [3]. In this approach context dependent clustering was used to train the kNN prototypes. Context dependent clustering is a family of grouping algorithms which use external, user defined variable (for each input vector) describing the strengths of association between the input vector and external parameter. Context clustering was studied by Pedrycz [31, 29], Łęski [36, 20] and others, and has been applied with very good results in training of the RBF networks [30, 1].

3.2 The conditional fuzzy clustering algorithm

One of methods that belong to the context dependent clustering family of algorithms is Conditional Fuzzy C-Means (CFCM). It is based on minimizing cost function defined as:

$$J_m(\mathbf{U},\mathbf{P}) = \sum_{i=1}^{c}\sum_{k=1}^{m} (u_{ik})^{\delta} \left\|\mathbf{x}_k - \mathbf{p}_i\right\|_A^2 \tag{29}$$

where c is the number of clusters centered at $\mathbf{p}_i$, m is the number of vectors, $\delta > 1$ is a parameter describing fuzziness, and $\mathbf{U} = (\mathbf{u_{ik}})$ is a $c \times m$ dimensional membership matrix with elements $u_{ik} \in [0,1]$ defining the degree of membership of the k-th vector in the i-th cluster. The matrix $\mathbf{U}$ has to fulfill three conditions:
1^o each vector x_k belongs to the i-th cluster to some degree:

$$\bigvee_{1\leq i\leq c}\bigvee_{1\leq k\leq m} u_{ik}\in[0,1] \tag{30}$$

2^o sum of the membership values of k-th vector x_k in all clusters is equal to f_k

$$\bigvee_{1\leq k\leq m}\sum_{i=1}^{c} u_{ik}=f_k \tag{31}$$

3^o no clusters are empty.

$$\bigvee_{1\leq i\leq c} 0<\sum_{k=1}^{m} u_{ik}<m \tag{32}$$

Under these conditions cost function (29) reaches minimum for [29], :

$$\bigvee_{1\leq i\leq c}\mathbf{p}_i=\sum_{k=1}^{m}(u_{ik})^{\delta}\mathbf{x}_k\left[\sum_{k=1}^{m}(u_{ik})^{\delta}\right]^{-1} \tag{33}$$

$$\bigvee_{\substack{1\leq i\leq c\\1\leq k\leq m}} u_{ik}=f_k\left[\sum_{j=1}^{c}\left(\frac{\|\mathbf{x}_k-\mathbf{p}_i\|}{\|\mathbf{x}_k-\mathbf{p}_j\|}\right)^{2/(\delta-1)}\right]^{-1} \tag{34}$$

3.3 Determining the context

In classification problems the goal is to find a small number of prototypes that define classification border. In simple cases when linear solution is sufficient one prototype far from decision border implements approximately linear threshold P-rule. In more complex situations prototypes that are close to the decision border are needed, and they are also close to vectors from the opposite classes. This leads to a conclusion that grouping algorithms should be focused on clusters found close to the decision border and not on the whole space. For the context dependent clustering appropriate coefficients $f(k)$ taking this into account should be defined. Such a coefficient can be introduced in various ways, with one possible approach [3] based on the ratio of distances:

$$w_k=\sum_{j,C(\mathbf{x}_j)=C(\mathbf{x}_k)}\|\mathbf{x}_k-\mathbf{x}_j\|^2\left[\sum_{l,C(\mathbf{x}_l)\neq C(\mathbf{x}_k)}\|\mathbf{x}_k-\mathbf{x}_l\|^2\right]^{-1} \tag{35}$$

These coefficients are renormalized to fit the [0,1] range:

$$w_k\longleftarrow\left(w_k-\min_i w_i\right)\left(\max_i w_i-\min_i w_i\right)^{-1} \tag{36}$$

Normalized w_k coefficients reach values close to 0 for vectors inside large homogeneous clusters, and close to 1 if the vector x_k is near the vectors of the opposite classes and far from other vectors from the same class (for example if it is an outlier). These normalized weights determine the external variable which then is used to assign appropriate context or condition in the CFCM clustering process.

$$f_k = \exp\left(-\eta(w_k - \mu)^2\right) \tag{37}$$

with the best parameters in the range of $\mu = 0.6-0.8$ and $\eta = 1-3$, determined empirically for a wide range of datasets. The μ parameter controls where the prototypes will be placed; for small μ they are closer to the center of the cluster and for larger μ closer to the decision borders. The range in which they are sought is determined by the η parameter.

3.4 Numerical illustration of the CFCM approach

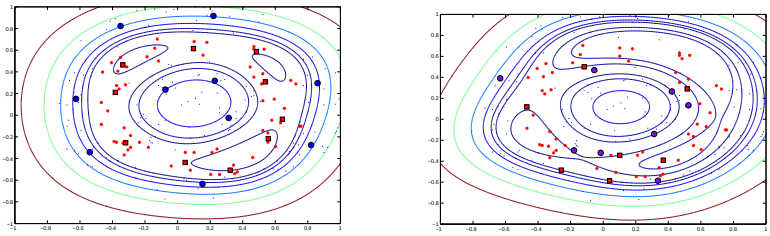
Conditional clustering proposed above does not use SVM to place prototypes directly, but the adjustment of weights is based on the SVM decision function. To verify this approach some simple numerical experiments were performed. Because in the CFCM method the number of prototypes for each class has to be determined independently the total number of desired SVs has been divided equally among the classes.

An artificial dataset example with a ring of data from one class between inner circle and outer data from another class was considered first, generated using the Spider toolbox subroutines [38]. Results from the Fixed Point Iteration calculations (Schölkopf algorithm) and from the CFCM-based algorithm described are presented in figure (5). The number of SVs was set to 20 (in CFCM 10 SVs per class), and Gaussian SVM parameters $C = 10000$ and $\sigma = 1$ have been used for both methods.

5 SVs found by the Fixed Point algorithm could not be plotted because they lie outside of the figure. Mean accuracy of the original SVM was $92.0 \pm 1.7\%$, for the Fixed Point algorithm 87.5 ± 1.4 and 91.5 ± 2.3 for the CFCM algorithm.

This example shows that our CFCM algorithm finds prototypes that are informative and represent the shape of the decision border with high accuracy. More knowledge can be derived from CFCM prototypes if the number of prototypes per class is optimized. This can be done using for example the racing algorithm described in [3].

Three well known benchmark datasets from the UCI repository [28] were used to verify quality of the proposed solution on real data. The Pima Indians Diabetes, Cleveland Heart Disease, and Ionosphere datasets have been selected. All calculations were performed with the Spider toolbox [38] using the 5-fold crossvalidation, extended by our own subroutines for CFCM prototype selection algorithm. In all cases the number of SVs were fixed to 4 (in CFCM two per class), the C value for SVM was optimized using the greedy search approach, and the Gaussian kernel parameter was fixed $\sigma = 1$. Classification results are presented in Table (1).



(a) SVs selected using CFCM clustering

(b) SVs selected using Schölkopf approach

Fig. 5. Comparison of the CFCM and Fixed Point algorithms on artificial data, showing support vector positions.

	#SV	SVM	Schölkopf RS-SVM	Burges RS-SVM	CFCM
Pima	305	23.4±2.3	37.9 ± 7.6	38.3 ± 8.1	25.9 ± 1.3
Cleveland	99	20.9±1.9	44.5 ± 8.8	31.6 ± 8.2	18.9 ± 1.8
Ionosphere	65	6.3±1.3	18.8 ± 3.3	16.5 ± 2.7	13.1 ± 1.9

Table 1. 5xCV classification error on the 3 datasets; the number of SVs from the Gaussian SVM given in the second column has been reduced in all cases to 4.

These results show that also on the real-word problems CFCM clustering combined with SVM gives quite good results. For such a small number of SVs Schölkopf and Burges RS-SVM algorithms give rather poor results, while RS-SVM based on CFCM clustering on the Cleveland Heart dataset obtained even better results then the original SVM classifier.

4 Conclusions

Although SVMs is a very powerful black box data classification tool it cannot be used directly for problems where decisions should be comprehensible. The hyperplane found in the image space cannot be easily translated into the knowledge useful for data understanding in the original feature space. Therefore various ways of expressing this knowledge should be studied. In this paper prototype-based rules are advocated as a natural extension of most of the rule based systems, well suited to the form of knowledge that may be derived from the SVM algorithm.

To represent knowledge contained in the SVM model in a comprehensible way as P-rules reduction of the number of SVs is necessary. This topic has been studied by

many experts and a few approaches have been discussed in this chapter. Minimization of the distance between original SVM hyperplane and the one obtained after reduction of the number of SVs does not seem to be correlated with the accuracy of the system obtained in this way. More comprehensible results are obtained using cost functions that are based directly on the classification accuracy.

Another problem that is facing P-rules based on typical RS-SVM algorithms is the interpretation of obtained prototypes. A solution proposed here is to use algorithms developed for optimization of the classical kNN classifiers. As an example conditional clustering algorithm (CFCM) was adopted to learn prototypes (SV) from the original dataset, with SVM hyperplane used to fit appropriate weights to selected prototypes. Results of such a combination on the artificial and real data used in this paper appear to be quite good, although it should be tested on much wider range of data and actual knowledge in form of P-rules should be carefully analyzed. This approach should be combined with feature selection to simplify further the interpretation of the rules.

An alternative to the rule-based understanding of the SVM function may be based on visualization techniques, as it has been done for MLP [10] and RBF neural networks [11, 9].

References

1. M. Blachnik. Warunkowe metody rozmytego grupowania w zastosowaniu do uczenia radialnych sieci neuronowych. Master's thesis, Silesian University of Technology, Gliwice, Poland, 2002.
2. M. Blachnik and W. Duch. Prototype-based threshold rules. *Springer Lecture Notes in Computer Science*, 4234, 2006.
3. M. Blachnik, W. Duch, and T. Wieczorek. Selection of prototypes rules – context searching via clustering. *Lecture Notes in Artificial Intelligence*, 4029:573–582, 2006.
4. C. Burges. Simplified support vector decision rules. In *International Conference on Machine Learning*, pages 71–77, 1996.
5. Y. Chen and J.Z. Wang. Support vector learning for fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 11(6):716–728, 2003.
6. R. Collobert and S Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.
7. T. Downs, K. Gates, and A. Masters. Exact simplification of support vector solutions. *The Journal of Machine Learning Research*, 2:293–297, 2001.
8. W. Duch. Similarity based methods: a general framework for classification, approximation and association. *Control and Cybernetics*, 29:937–968, 2000.
9. W. Duch. Coloring black boxes: visualization of neural network decisions. In *Int. Joint Conf. on Neural Networks, Portland, Oregon*, volume I, pages 1735–1740. IEEE Press, 2003.
10. W. Duch. Visualization of hidden node activity in neural networks: I. visualization methods. In L. Rutkowski, J. Siekemann, R. Tadeusiewicz, and L. Zadeh, editors, *Lecture Notes in Artificial Intelligence*, volume 3070, pages 38–43. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2004.

11. W. Duch. Visualization of hidden node activity in neural networks: Ii. application to rbf networks. In L. Rutkowski, J. Siekemann, R. Tadeusiewicz, and L. Zadeh, editors, *Lecture Notes in Artificial Intelligence*, volume 3070, pages 44–49. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2004.
12. W. Duch, R. Adamczak, and G. H. F. Diercksen. Distance-based multilayer perceptrons. In M. Mohammadian, editor, *International Conference on Computational Intelligence for Modelling Control and Automation*, pages 75–80, Amsterdam, The Netherlands, 1999. IOS Press.
13. W. Duch, R. Adamczak, and G.H.F. Diercksen. Classification, association and pattern completion using neural similarity based methods. *Applied Mathemathics and Computer Science*, 10:101–120, 2000.
14. W. Duch, R. Adamczak, and K. Grąbczewski. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12:277–306, 2001.
15. W. Duch and M. Blachnik. Fuzzy rule-based systems derived from similarity to prototypes. In N.R. Pal, N. Kasabov, R.K. Mudi, S. Pal, and S.K. Parui, editors, *Lecture Notes in Computer Science*, volume 3316, pages 912–917. Physica Verlag, Springer, New York, 2004.
16. W. Duch and G. H. F. Diercksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87:341–371, 1995.
17. W. Duch and K. Grudziński. Prototype based rules - new way to understand the data. In *IEEE International Joint Conference on Neural Networks*, pages 1858–1863, Washington D.C, 2001. IEEE Press.
18. W. Duch and K. Grudziński. Meta-learning via search combined with parameter optimization. In L. Rutkowski and J. Kacprzyk, editors, *Advances in Soft Computing*, pages 13–22. Physica Verlag, Springer, New York, 2002.
19. W. Duch, R. Setiono, and J. Zurada. Computational intelligence methods for understanding of data. *Proceedings of the IEEE*, 92(5):771–805, 2004.
20. J. Łęski. Ordered weighted generalized conditional possibilistic clustering. In J. Chojcan and J. Łęki, editors, *Zbiory rozmyte i ich zastosowania*, pages 469–479. Wydawnictwa Politechniki Śląskiej, Gliwice, 2001.
21. K. Grąbczewski and W. Duch. Heterogeneous forests of decision trees. *Springer Lecture Notes in Computer Science*, 2415:504–509, 2002.
22. M. Grochowski and N. Jankowski. Comparison of instance selection algorithms. ii. results and comments. *Lecture Notes in Computer Science*, 3070:580–585, 2004.
23. N. Jankowski and M. Grochowski. Comparison of instance selection algorithms. i. algorithms survey. *Lecture Notes in Computer Science*, 3070:598–603, 2004.
24. T. Joachims. *Learning to Classify Text Using Support Vector Machines*. Kluwer Academic Publisher, 2002.
25. L.I. Kuncheva and J.C. Bezdek. An integrated framework for generalized nearest prototype classifier design. *International Journal of Uncertainty*, 6(5):437–457, 1998.
26. J.T. Kwok and I.W. Tsang. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15:408–415, 2003.
27. K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14(6):1449–1459, 2003.
28. C.J. Merz and P.M. Murphy. UCI repository of machine learning databases, 1998-2004. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
29. W. Pedrycz. Conditional fuzzy c-means. *Pattern Recognition Letters*, 17:625–632, 1996.
30. W. Pedrycz. Conditional fuzzy clustering in the design of radial basis function neural networks. *IEEE Transactions on Neural Networks*, 9(4), 1998.

31. W. Pedrycz. Fuzzy set technology in knowledge discover. *Fuzzy Sets and Systems*, 98(3):279–290, 1998.
32. E. Pękalska and R.P.W. Duin. *The dissimilarity representation for pattern recognition: foundations and applications*. New Jersey; London: World Scientific, 2005.
33. J. Platt. Using sparseness and analytic qp to speed training of support vector machines. *Advances in Neural Information Processing Systems*, 11, 1999.
34. I. Roth and V. Bruce. *Perception and Representation*. Open University Press, 1995. 2nd ed.
35. B. Schölkopf, P. Knirsch, A. Smola, and C. Burges. Fast approximation of support vector kernel expansions. *Informatik Aktuell, Mustererkennung*, 1998.
36. J. Łęski. A new generalized weighted conditional fuzzy clustering. *BUSEFAL*, 81:8–16, 2000.
37. T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to model ing and control. *IEEE Transactions on Systems, Man, Cybernetics*, 15:116–132, 1985.
38. J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The spider. http://www.kyb.tuebingen.mpg.de/bs/people/spider/.
39. D.R. Wilson and T.R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38:257–268, 2000.