with

$$W\left(\mathbf{X}|\mathrm{d}^v\right) = \sum_{k=1}^{C} \sum_{n_1=1}^{N_k-1} \sum_{n_2=n_1+1}^{N_k} \mathrm{d}^v\left(\boldsymbol{x}_{k,n_1}, \boldsymbol{x}_{k,n_2}|\boldsymbol{\lambda}\right) ,$$

$$B\left(\mathbf{X}|\mathrm{d}^v\right) = \sum_{k_1=1}^{C-1} \sum_{k_2=k_1+1}^{C} \sum_{n_1=1}^{N_{k_1}} \sum_{n_2=1}^{N_{k_2}} \mathrm{d}^v\left(\boldsymbol{x}_{k_1,n_1}, \boldsymbol{x}_{k_2,n_2}|\boldsymbol{\lambda}\right) .$$

This ratio involves within-class compression $W\left(\mathbf{X}|\mathrm{d}^v\right)$ and between-class expansion $B\left(\mathbf{X}|\mathrm{d}^v\right)$ in terms of an adaptive rank-1 version of the Mahalanobis metric

$$\mathrm{d}^v\left(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\lambda}\right) = \left(\boldsymbol{x}-\boldsymbol{y}\right)^{\mathsf{T}} \cdot \boldsymbol{\Lambda} \cdot \left(\boldsymbol{x}-\boldsymbol{y}\right) = \left(\sum_{m=1}^{M} \lambda_m \cdot \left(x_m - y_m\right)\right)^2 . \tag{2}$$

Instead of a full rank matrix, the outer self-product of the parameter vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)^{\mathsf{T}}$ defines an adaptive matrix $\boldsymbol{\Lambda} = \boldsymbol{\lambda} \cdot \boldsymbol{\lambda}^{\mathsf{T}}$ of rank one. This positive-definite matrix contains components that weigh the influence of attribute pairs (i,j) along a class-separating direction $\boldsymbol{\lambda}$ in the data space.

For minimizing the stress function in Eqn. 1 in an efficient manner, we make use of the memory-limited quasi-Newton method [6]. For such gradient-based optimization approach we provide the derivative of Eqn. 2:

$$\frac{\partial \mathrm{d}^v\left(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\lambda}\right)}{\partial \lambda_i} = 2 \cdot \left(x_i - y_i\right) \cdot \sum_{m=1}^{M} \lambda_m \cdot \left(x_m - y_m\right) . \tag{3}$$

Optimization is stopped, if the improvement of subsequent stress function evaluations falls below 10^{-8}. The SARDUX algorithm makes use of the optimization routines from the Java framework for statistical analysis and classification of biological sequences (`http://www.jstacs.de`) and is available at the web site `http://dig.ipk-gatersleben.de`.

2.2 Feature selection workflow

The SARDUX algorithm requires data to be assigned to classes. For that purpose, the k-means clustering algorithm with $k = 2$ is applied to the logP values to partition the associated data into two classes. A number of 100 random starts with 1500 iterations ensures a stable k-means partitioning. From the clusters with minimum quantization error, a threshold of $\mathrm{logP}_* = 2.61$ is determined as the mean value of the minimum of compounds connected to logP values of the high logP centroid and the maximum of compounds connected to the low centroid. This way, roughly 57% of compounds are assigned to logP values less or equal to logP_*, constituting class 0, and 43% of high logP compounds constitute class 1. This coarse data partitioning into only two-classes has got the advantage that a single direction vector between two unimodal classes can be uniquely determined later on.

After partitioning, SARDUX is applied 100 times with random initializations of λ_i on random data splits into 2:1 of training and validation samples. Thereby, each split is created by stratified sampling to maintain the portion of 57:43 for the class assignments. In order to get independent of individual data splits and initializations, an average model is created from the top 10 converged out of the 100 runs. Thereby, performance is measured as class confusion of the data after projection to the direction vector of model. For any projected data point, being just a real value, the fraction of points with even smaller (larger) values belonging to the other class is computed. The best class separation is at a confusion value of 0%, i.e., all values of the other class are systematically located on one side either less or greater than that value. The average total class confusion of test and validation data is used for model assessment.

The 10 models with smallest confusion scores are selected, and each direction vector $\boldsymbol{\lambda}$ is turned into a rank vector $\boldsymbol{\gamma} = \mathrm{rnk}(|\boldsymbol{\lambda}|)$, thereby, considering absolute vector entries invariant to direction. The sum of such ranks for each attribute robustly quantifies its contribution to the class separation: the higher the value, the higher the influence. Thus, if $\boldsymbol{\gamma}$ denotes the average rank vector, then $\boldsymbol{\Gamma} = \boldsymbol{\gamma} \cdot \boldsymbol{\gamma}^{\mathsf{T}}$ is the corresponding attribute dependence matrix.

An intensity-based visualization of the matrix provides first hints about important attributes with many strong connection to other attributes. A more systematic assessment is possible by applying hierarchical clustering with optimum leaf ordering to the rows and columns of the matrix [2]. In the present case of only one available separating direction the structure of matrix $\boldsymbol{\Gamma}$ of rank 1 is rather simple and allows a nice ordering for feature selection. Finally, from such a reordered matrix, groups of interesting attributes can be easily identified.

2.3 Regression by neural networks with one hidden layer

Feed-forward artificial neural networks are universal function approximators [8] with frequent use in regression tasks such as ADMET prediction [10,16]. Here, prediction is based on network architectures with one hidden layer trained by a modification of resilient back propagation [1]. For comparability with the first publication in the area [16], architectures with one hidden layer of 11 neurons and a single output neuron are considered. Generally, a fixed internal network architecture might be considered as design flaw, but rather than providing perfectly optimized regression models it increases comparability and saves computing time while providing insights into effects of feature subset selection.

All computer experiments are realized in the freely available R/Bioconductor package for statistical computing [3,7]. The R-package 'neuralnet' has been modified to stop the training process after 1 million iterations, i.e., at asymptotic convergence. This modification circumvents the problem of the original implementation that requires choosing an error-based termination threshold biased by architecture and data structure. The desirable option of stopping at error increase on a validation set is by now not available in that package. For each considered feature subset scenario the network training is repeated 10 times with random initialization, and the lowest error run is selected for the prediction of

the test samples. Generally, repetition numbers like 10, 100 or 1 million times were chosen to find a compromise of computing time and optimum accuracy.

The mean average error (mae) per compound prediction is the performance measure of interest. Only for comparison with the reference results [16] squared correlation of predicted and observed logP values are given. In general, high correlation does not mean too much because it does neither account for a systematic scaling bias nor for outliers in the average range of values.

2.4 Data preparation

A compound data set with 12 molecular features and associated logP values for 440 chemical compounds has been taken from [16] for the analysis. Due to duplication of di-n-propyl-amine its second occurrence, i.e., compound number 277, has been deleted, leaving 439 records for the subsequent analysis. By database queries 61 additional molecular features were included for these compounds, providing a data set with compounds $\mathbf{x}^j$ described by $M = 73$ features. The original data set comes with a fixed split into training/validation sets and a test set. An initial study showed that the test set data are completely contained within the convex hull of the training/validation data, which does not allow an unbiased assessment of the generalization properties of a regression model. Furthermore, the test data were normalized together with the training and validation data, which is undesirable, because in real-life scenarios new data must be considered as unavailable during normalization, i.e., before the model creation.

Here, an independent test set of 30 compounds has been created that covers the range of logP values uniformly and that is not confined in the convex hull of the remaining data. These test set indices are 3, 7, 13, 100, 121, 135, 151, 192, 194, 203, 211, 224, 254, 262, 263, 291, 294, 320, 321, 352, 363, 376, 378, 380, 383, 384, 386, 398, 419, and 426. On the remaining data the range $[l_i^{\min}\, l_i^{\max}]$ of each attribute $i = 1 \ldots 73$ was calculated. These ranges were applied to the whole data set as $\mathbf{x}_i^j \leftarrow (\mathbf{x}_i^j - l_i^{\min})/(l_i^{\max} - l_i^{\min})$, $j = 1 \ldots 439$. The training/validation set is strictly within the unit hypercube, while the test data and any new compound contain a number of attributes slightly outside the unit interval, except for those with binary states.

3 Results

Neural networks are suitable for regression. Before any feature selection tasks was addressed, neural networks were tested on the original training and test data from [16]. The particular implementation of the ARTMAP regression model introduced therein was not available. Therefore we sticked to the more common feed-forward network architecture for studying the effects of feature selection. The R-package 'neuralnet' provides easy-to-use network training using of a resilient propagation. The reference results of $r^2 = 0.94$ and mae $= 0.27$ for the training/validation set and $r^2 = 0.96$ and mae $= 0.23$ for the test set could be improved by the 'neuralnet' package to $r^2 = 0.960$ and mae $= 0.206$ for

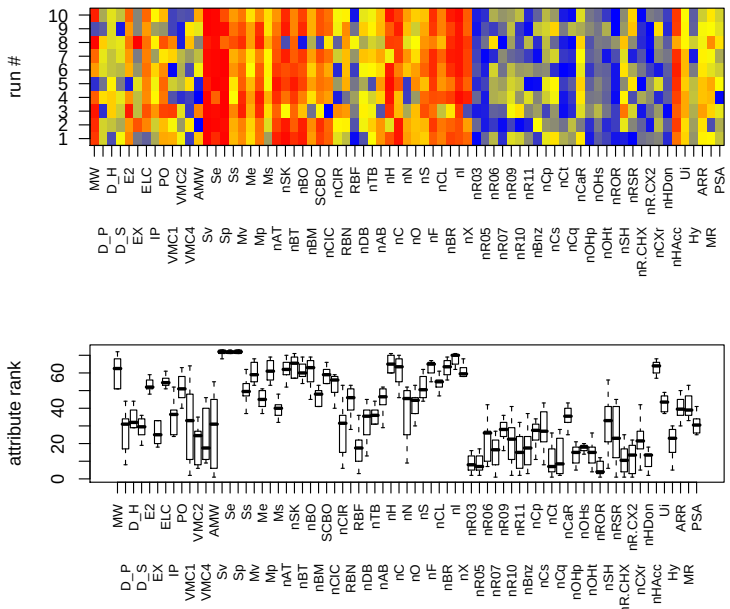

Fig. 1. SARDUX model selection (top) and average model (bottom) for the assessment of molecular descriptors related to the logP value. Runs in the top panel are ordered from 1 to 10 according to increasing class confusion. Blue, yellow, and red colors denote low, medium, and high feature importance, respectively, pointing out good reproducibility across different data splits and training runs. The bottom panel contains boxplots of the attribute ranks of these 10 runs. Higher ranks indicate higher contribution to the attribute weight matrix.

the training/validation set and $r^2 = 0.964$ and mae $= 0.211$ for the test set. Although these results are still inferior to those for fuzzy ARTMAP reported in [16], they demonstrate a good reliability for our regression tasks since a more rigorous validation procedure is applied here.

SARDUX feature weight model. As described in the methods section, 10 out of 100 SARDUX runs with minimum class confusion are selected for creating a robust average feature ranking model. The optimization of all 100 models took roughly 5h on a standard dual core machine, i.e., 3 minutes per model. Results of the top 10 runs are shown in the top panel of Figure 1 as color rows, displaying the ranks of absolute entries in each model vector $\boldsymbol{\lambda}$, and a boxplot of these models is shown in the bottom panel.

Linear regression supports attribute relevance. The outcome of the model with minimum class confusion is shown in Figure 2. The percent of con-

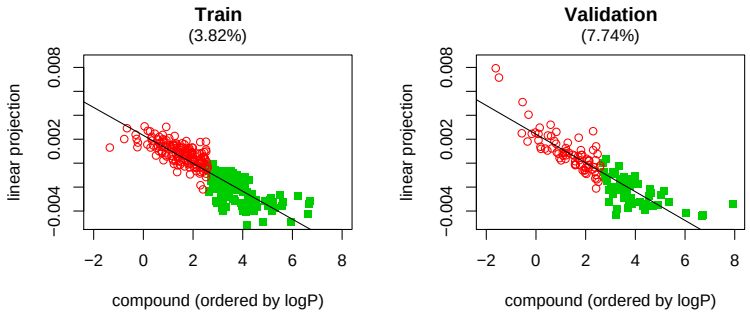

Fig. 2. Linear projection of the training data (left) and validation data (right) onto the SARDUX direction vector $\boldsymbol{\lambda}$ resulting in minimum class confusion. Data are ordered from left to right in increasing order of associated logP value; values below 2.61 correspond to class 0 (red points), larger values belong to class 1 (green points). Linear regression lines are provided for reference. Percent values in parentheses denote the degree of class confusion.

fusion is rather small for the training set and it does not degrade too much for the validation data. One important observation can be made by help of the regression lines: although only a two-class separation is optimized, the obtained projection vector contains information for creating a tentative linear regression model between the projected attributes and the logP value. In other words: the feature weights contain useful information for a prediction model.

Feature selection from the SARDUX weight matrix. The bottom panel of Figure 1 is an averaged model of absolute attribute contributions $\boldsymbol{\gamma}$. This must not be confused with the final model of ranked attributes relating back to the matrix metric as $\boldsymbol{\Gamma} = \boldsymbol{\gamma} \cdot \boldsymbol{\gamma}^{\mathsf{T}}$. That matrix is shown in the left panel of Figure 3. In this matrix context the sum of all contribution pairs with each attribute represents the overall relevance of that attribute. Although it is tempting to provide an ordering of the attributes along decreasing values of these row and column sums, one general aspect must be considered with care. Large sums might arise in two cases: either because only some other attribute provide a few very large values for an attribute, or because one attribute interacts with many others on a low level. For strict feature selection, specific dependence on few other attributes is of interest, because generally dependent attributes lose substantially in combination with only a fraction of the original features. For matrices of rank one its structure is rather simple and the result of optimum leaf ordering displayed in the right panel of Figure 3 is equivalent to the ordering of summed up attribute ranks. Although this ordering is similar with the ordering of the average rank of the projection vector components in the lower panel of Figure 1, such as for Sp, Se, and Sv, there are also differences, such as nSK possessing ranks 14 and 5, respectively.

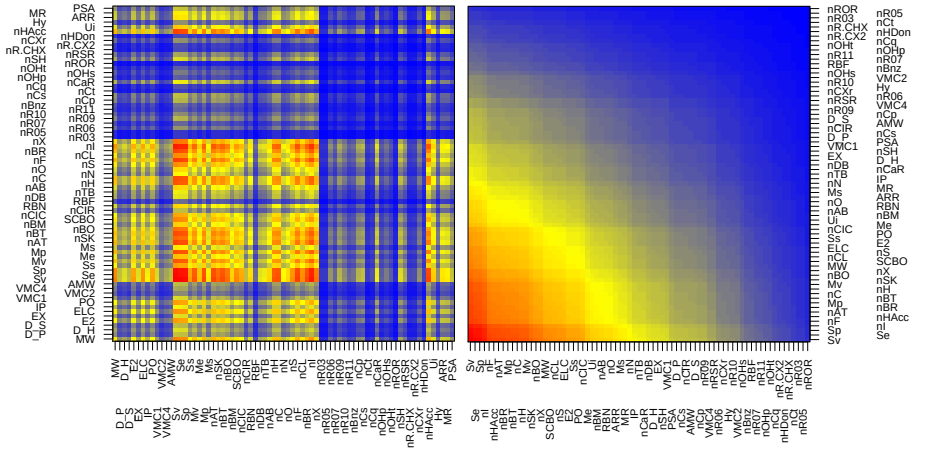

Fig. 3. SARDUX attribute relevance matrices $\boldsymbol{\Gamma}$. Blue, yellow, and red colors, respectively, denote low, medium, and great importance of feature pairs. Left: attributes in the original order. Right: attributes reordered by optimum leaf ordering applied to an average link hierarchical clustering of the rows and columns of the left matrix with decreasing order of relevance from left to right.

Feature selection performance. For comparison with the reference model from [16] the results of the top 12 SARDUX attributes are compiled in blocks A and B of Table 1. Each network selection takes about 4 hours to complete 10 randomly initialized training runs with 1 million iterations. First of all, the two attribute subsets are completely disjoint, yet, non-differentiable mae performance is obtained on the test set. Scatter plots of prediction and observation of the logP values are shown in Figure 4.

Since it seems surprising to find completely disjoint subsets with similar performance, we studied the performance of 25 randomly selected feature subsets, where each subset has 12 features randomly sampled from the total of 73 features. This required 100 hours of computing time. Two out of these 25 feature subsets led to even better prediction models than both SARDUX and the reference. The average performance, though, reported in section D* of Table 1 indicates a clear superiority of targeted feature selection.

A random model with average performance representing Section D* of Table 1 is displayed in the lower right panel of Figure 4. The best random subset is reported in section D of Table 1. It improves the test mean average error by 0.13, i.e. 2.7%, compared to the SARDUX model and by 0.17, i.e. 3.5%, compared to the reference. Only the feature nAT is shared with SARDUX, but the features D_P, ELC, and VMC1 are shared with the reference set.

For assessing general attribute correlations, the dissimilarity matrix of (1-Pearson correlation)^8 between the 73 attributes of the training set has been embedded by a multidimensional scaling approach [13] into the 2D plane shown in Figure 5. In that figure highly correlating features are grouped together, such

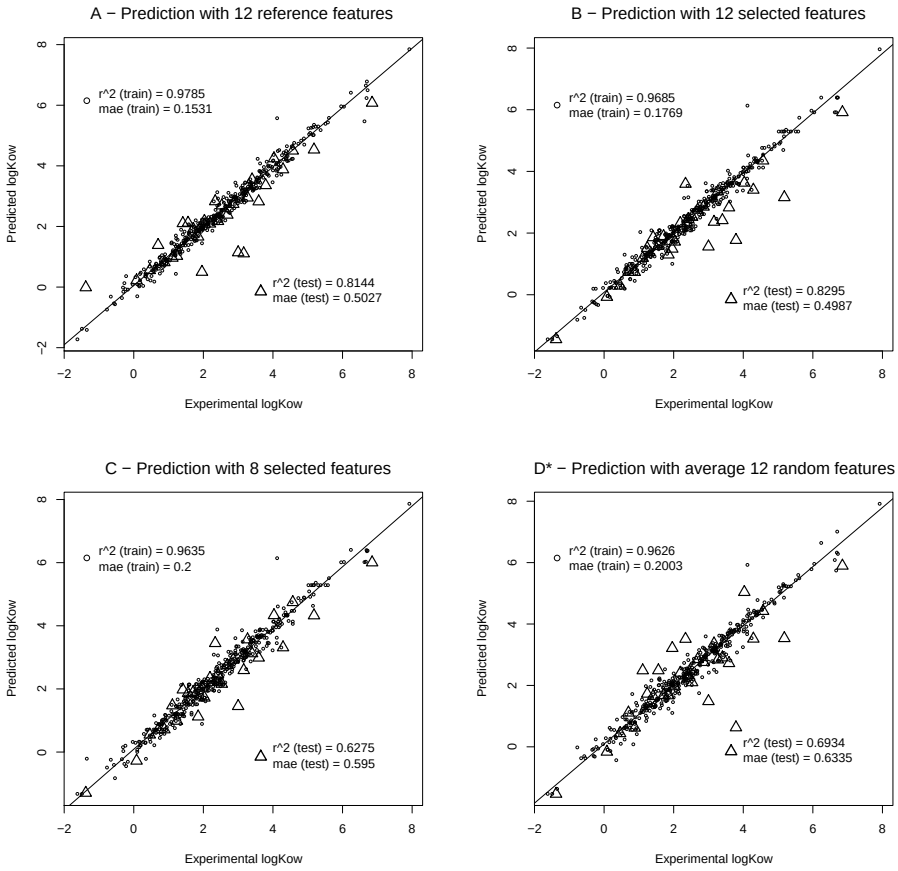

Fig. 4. Scatter plots of neural network regressions for section A, B, C, and D* of Table 1.

as Sp and Sv, D_S and D_P or VMC2 and VMC4. Although it does not reveal relevance, such a display helps to identify not only redundant features but also singular ones like nH, RBF, IP, EX, and Ms.

By looking at the right panel of Figure 3 a block structure can be observed, i.e., redundant features with similar dependence pairs are located within one block. Only those features not belonging to the same block, but with sufficient overall contribution, have been selected, resulting in a set of 8 manually selected attributes compiled in section C of Table 1 and in the lower left scatter plot of Figure 4. Although the training performance is still acceptable, the test performance is dramatically deteriorated. This means that is better to stick to redundant but relevant features for the network training.

Top features. In decreasing order of relevance for blocks **B** and **C**.											
1	2	3	4	5	6	7	8	9	10	11	12
A	Reference features										
MW	D_P	D_H	D_S	E2	EX	ELC	IP	PO	VMC_1	VMC_2	VMC_4
train:		$r^2 = 0.979$		mae=0.153		test:		$r^2 = 0.814$		mae=0.503	
B	Top selected features using SARDUX										
Sv	Se	Sp	nI	nF	$nHAcc$	nAT	nBR	Mp	nBT	nC	nH
train:		$r^2 = 0.969$		mae=0.177		test:		$r^2 = 0.830$		mae=0.499	
C	8 manually selected features assisted by SARDUX										
Sv	nI	nF	nH	nSK	MW	E2	nCIC	-	-	-	-
train:		$r^2 = 0.964$		mae=0.200		test:		$r^2 = 0.628$		mae=0.600	
D	Best 12 features out of 25 randomly selected subsets										
D_P	ELC	Hy	nAT	nCq	nDB	nN	$nRCX_2$	nR03	nR06	RBN	VMC_1
train:		r^2=0.971		mae=0.174		test:		$r^2 = 0.870$		mae=0.486	
D*	Average performance of 25 randomly selected subsets										
train:		$r^2 = 0.961 \pm 0.022$				test:		$r^2 = 0.681 \pm 0.143$			
		mae=0.200 ± 0.0484						mae=0.652 ± 0.140			

Table 1. Results of different feature selection strategies.

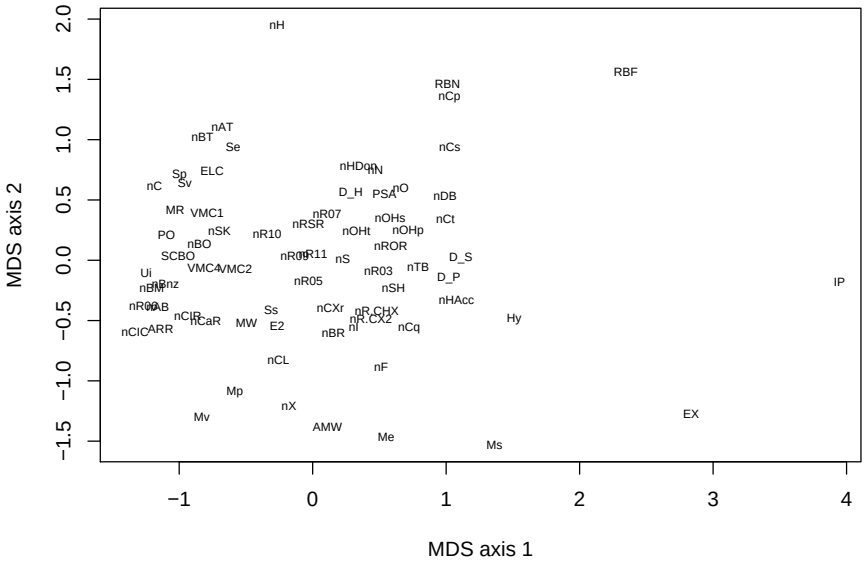

Fig. 5. Multidimensional scaling of the correlation between features.

3.1 Discussion

The fact of random selection getting sometimes better than a systematic one is not a disappointing result. Firstly, the computational requirements of SARDUX feature selection against expensive neural network training are very substantial. Selecting the best out of 100 SARDUX models, takes only 5 instead of 100 hours, without the need to train a single neural network. Secondly, if at least 12 features are considered to be meaningful in the data set and since there is some degree of redundancy, then a random subset of 12 features will contain some of the important or their collinear features just by chance. The probability of a randomly good choice is rather high, especially if the feature quality for the prediction is at similar levels. Thirdly, the full potential of SARDUX is not yet exploited, because currently only one direction, i.e., a simple-structured matrix metric, is involved. If in future developments SARDUX gets extended to several unique and differentiating directions, the predictive feature quality will automatically increase.

4 Conclusions

In contrast to traditional feature assessment methods, the proposed adaptive matrix metric contains information not only about singular attributes, but about pairs of attributes. This is particularly useful for neural networks, because they integrate over all possible feature combinations in the hidden layer. The presented SARDUX method is currently limited in two ways: (i) only one direction can be identified, and (ii) a class-related problem is optimized rather than a regression problem. Yet, as shown by linear projection onto the mostly class-separating direction, the projection value shows rough linear dependence on the logP value.

The study has shown that completely disjoint feature subsets for the proposed way and the reference lead to almost identical performance of the regression models. These models are much better than the average random feature selection model, but worse than the two best random selections. Whether or not those good random subsets are due to redundancies in the data set, the results of SARDUX-based feature selection are encouraging enough to address the two limitations reported above in future work.

This work is kindly supported by BMBF grant ARG 08/016 and by grant XP3624HP/0606T of the Ministry of Culture Saxony-Anhalt. Also we thank MINCyT for his grant AL0811 and UNS for grant PGI 24/ZN16.

References

1. A. Anastasiadis, G. Magoulas, and M. Vrahatis. New globally convergent training scheme based on the resilient propagation algorithm. *Neurocomputing*, 64:253–270, 2005.
2. Z. Bar-Joseph, D. Gifford, and T. Jaakkola. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics*, 17(Suppl. 1):S22–29, 2001.
3. R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biol.*, 5:R80, 2004.
4. B. Hammer, M. Strickert, and T. Villmann. Relevance LVQ versus SVM. In L. Rutkowski, J. Siekmann, R. Tadeusiewicz, and L. Zadeh, editors, *Artificial Intelligence and Softcomputing*, volume 3070 of *LNAI*, pages 592–597. Springer, 2004.
5. G. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley-Interscience, 2004.
6. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, August 1999.
7. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008.
8. R. Rojas. *Neural Networks - A Systematic Introduction*. Springer, Berlin, 1996.
9. P. Schneider, M. Biehl, and B. Hammer. Matrix adaptation in discriminative vector quantization. IfI Technical Report Online IFI-08-08, Clausthal University of Technology, 2008.
10. A. Soto, R. Cecchini, G. Vazquez, and I. Ponzoni. A Wrapper-Based Feature Selection Method for ADMET Prediction Using Evolutionary Computing. In E. Marchiori and J. Moore, editors, *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, LNCS 4973*, pages 188–199. Springer, 2008.
11. A. Soto, R. Cecchini, G. Vazquez, and I. Ponzoni. An evolutionary approach for feature selection applied to ADMET prediction. *Inteligencia Artificial*, 12:55–63, 2008.
12. M. Strickert, P. Schneider, J. Keilwagen, T. Villmann, M. Biehl, and B. Hammer. Discriminatory data mapping by matrix-based supervised learning metrics. In L. Prevost, S. Marinai, and F. Schwenker, editors, *Artificial Neural Networks in Pattern Recognition, LNCS 5065*, pages 78–89. Springer, 2008.
13. M. Strickert, N. Sreenivasulu, B. Usadel, and U. Seiffert. Correlation-maximizing surrogate gene space for visual mining of gene expression patterns in developing barley endosperm tissue. *BMC Bioinformatics*, 8(165), 2007.
14. Y. Sun. Iterative relief for feature weighting: Algorithms, theories, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1035–1051, 2007.
15. K. Weinberger and L. Saul. Fast solvers and efficient implementations for distance metric learning. In A. McCallum and S. Roweis, editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 1160–1167. Omnipress, 2008.
16. D. Yaffe, Y. Cohen, G. Espinosa, A. Arenas, and F. Giralt. Fuzzy ARTMAP and back-propagation neural networks based quantitative structure-property relationships (QSPRs) for octanol-water partition coefficient of organic compounds. *J. Chem. Inf. Comput. Sci.*, 42(2):162–183, 2002.
17. L. Yang. Distance Metric Learning: A Comprehensive Survey. Report, Michigan State University, http://www.cse.msu.edu/~yangliu1/, 2006.