Generalizing, Optimizing, and Decoding Support Vector Machine Classification*

Mario Michael Krell[1], Sirko Straube[1], Hendrik Wöhrle[2], and Frank Kirchner[1,2]

[1] University of Bremen, Faculty 3 – Mathematics and Computer Science, Robotics Lab, Robert-Hooke-Str.1, 28359 Bremen, Germany,
krell@uni-bremen.de,
[2] German Research Center for Artificial Intelligence, DFKI Bremen, Robotics Innovation Center, Robert-Hooke-Str. 1, 28359 Bremen, Germany

Abstract. A major challenge in the classification of complex data, that requires the combination of several processing steps, is the selection of the *optimal* algorithms for preprocessing and classification. Here, we present three steps to face this problem. First, we introduce a *generalized* model for Support Vector Machine (SVM) variants which generates both unary and online classifiers. This model improves the understanding of relationships between the variants which facilitates the choice and implementation of the classifier. Second, we propose the signal processing and classification environment pySPACE which enables the systematic evaluation and comparison of algorithms. Third, we introduce an approach called backtransformation which enables a visualization of the complete processing chain in the the input data space and thereby allows for a joint interpretation of preprocessing and classification to *decode* the decision process. Finally, the benefit of combining all three approaches is shown in an application on handwritten digit classification.

Keywords: pySPACE, support vector machine, relative margin, zero separation approach, online learning, backtransformation

1 Introduction

Dealing with classification tasks of complex spatiotemporal data like the electroencephalogram (EEG) one major issue lies in the generation of meaningful features. This is due to the fact that the data often consists of a superposition of a multitude of signals, together with dynamic, and observational noise. Hence, the data processing usually requires the combination of different preprocessing steps additionally to a classifier. In fact, the generation of good features is often more important than the actual classification algorithm [1]. Consequently, in many cases expert knowledge is required in order to specify the data processing. Furthermore, there is a very large number of processing algorithms and the interplay between them is often hard to grasp. Altogether, this makes it difficult

* This work was supported by the *German Federal Ministry of Economics and Technology* (BMWi, grants FKZ 50 RA 1012 and FKZ 50 RA 1011).

to automize the process of optimizing the data processing chain to get the best preprocessing and classification. In this paper, we present three related tools to make this process easier.

Due to the ever-growing number of classification algorithms, it is difficult to decide which ones to consider. Knowledge about the relations between the classifiers facilitates the choice and implementation of classifiers. As such, instead of further specializing existing classifiers we take a unifying view. Considering only the variants of the SVM [2–5] we developed the following general concepts building connections between these classifiers. The first concept, called relative margin [6, 7], enables a connection of SVM and regularized Fisher's linear discriminant (RFLD) [8]. The second concept, the zero separation approach, allows to define unary classifiers with the help of binary classifiers by taking the origin as a second class. Third, the single iteration approach transfers batch learning classifiers to online classifiers. If the batch algorithm is repeatedly iterating over the training samples to update a linear classification function, an online learning algorithm can be generated by performing this update only once with each incoming sample. Knowing these connections simplifies the implementation of the algorithms and makes it possible to transfer extensions or modifications from one algorithm to the other connected ones. Thus, it enables to build a classifier that fits into the individual research aims.

Nevertheless, it still required to optimize the hyperparameters and the preprocessing. Hence, it is necessary to have "an infrastructure that makes experimenting with many different learners, data sources, and learning problems easy and efficient" [1]. To solve this problem, we developed the signal processing and classification environment pySPACE [9]. It provides functionality for a systematic and automated comparison of numerous algorithms and parameterizations in a signal processing chain. Additionally, pySPACE enables the visualization of data, algorithms, and evaluation results in a common framework.

Optimizing the processing and knowing the relations between classifiers is not sufficient. It is also important to *understand* the final processing model to find out what lies behind the data. A first step is to visualize the data and the single processing steps, but this might not be sufficient for a complete picture, especially when dimensionality reduction algorithms are used in the preprocessing. This is quite often the case for high-dimensional and noisy data. Hence, a representation of the entire processing chain including *both* classification and preprocessing is required. Our approach to calculate this representation is called backtransformation. It iteratively transforms the classification function back through the signal processing chain to generate a representation in the same format as the input data. This representation provides weights for each part of the data to tell which components are relevant for the complete processing and which parts are ignored. It can be directly visualized using classical data visualization approaches as they are used for visualizations of images, EEG and functional magnetic resonance imaging (fMRI) data. This visualization can then be used to support the "close collaboration between machine learning experts and application domain ones" [1]. This can help to improve the processing

and to generate new knowledge about the data. In some cases even new expert knowledge might be generated.

In the following sections, we present our steps to improve and automatize the process of designing a good processing chain for a classification problem (classifier connections, pySPACE, backtransformation) including the related work in the respective area. We conclude by giving an application example which combines the three approaches in a unified approach.

2 Generalization: Classifier Connections

The classical SVM (C-SVM) is motivated by the concept of maximizing the distance between two hyperplanes, which separate positive from negative samples. This type of regularization is extended with a loss term, which allows for samples on the opposite side of these hyperplanes. Furthermore, lifting the data into a higher-dimensional space to make it linearly separable can be replaced with kernels. Only the scalar product of two samples is substituted by a kernel function. These powerful ideas and good performance results make the SVM attractive for numerous *variants*. Some examples are: support vector regression (SVR) [10], relative margin machines (RMMs) [6, 7], least squares SVM (LS-SVM) [11], ν-SVM [12], one-class SVM (νoc-SVM) [13], support vector data description (SVDD) [14], and passive-aggressive perceptrons (PAPs) [15]. Furthermore, RFLD can be seen as an SVM variant, too [7, 8]. Some connections between these classifiers are known. In the following sections, general concepts for a unifying view are proposed to connect these classifiers and ease the process of choosing a fitting classifier: relative margin, zero separation approach, and single iteration approach. They can generate a large number of additional variants (see Fig. 1).

2.1 Relative Margin

The relative margin concept [6] adds two additional (outer) parallel hyperplanes to the C-SVM model with a relative distance (range R) to the decision hyperplane. Relative distance means that the real distance is R times $\frac{1}{\|w\|}$, when w is the classification vector. Note, $\frac{2}{\|w\|}$ is the distance between the aforementioned maximum margin hyperplanes. If outliers at the new outer margin are treated in the same way as in the inner margin, the model is called balanced relative margin machine (BRMM) [7]. This model is equivalent to SVR (with the dependent variables $Y = \{-1, 1\}$) and connects SVM ($R = \infty$) and RFLD classification ($R = 1$) [7]. A squared loss function, kernels, and implementation techniques can be directly transferred from C-SVM to BRMM. BRMM has two hyperparameters: the range R and the C-SVM complexity parameter C. For optimization it is efficient to start with high values and iteratively decrease the values with a pattern search algorithm [16]. To save resources, the *warm start* principle can be used, to adapt the batch learning algorithms to the changed parameters [17]. With this parameter optimization, it is no longer necessary to choose between SVM and RFLD.

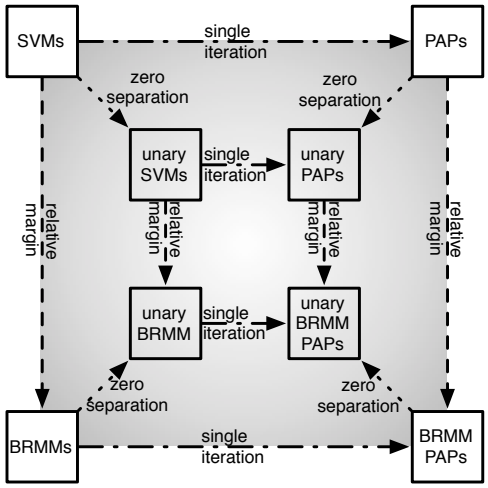


Fig. 1: Combining the approaches, introduced in Sec. 2: relative margin (vertical arrows) to generate the balanced relative margin machine (BRMM) which is the connection to the regularized Fisher's linear discriminant (RFLD), single iteration (horizontal arrows) to generate online classifiers like the passive-aggressive perceptron (PAP), and the zero separation (perpendicular arrows) to generate unary classifiers from binary ones.

2.2 Zero Separation Approach

In some applications, a second class is not of interest [18] or not enough examples of this class can be given as in outlier and novelty detection [19]. Hence, unary classifiers are required. In the zero separation approach a binary classifier is transferred to an unary classifier. The origin (zero) is added as a sample of the opposite (negative) class to the training data and the respective binary classifier is applied [13, 20]. The application of this concept to ν-SVM results in νoc-SVM, but it can be also applied to other classifiers like BRMM. Implementation techniques of the original model can be directly used.

2.3 Single Iteration Approach

The C-SVM is traditionally solved with sequential minimal optimization [21] as implemented in the LibSVM [22]. In the linear case, there are simplifications where the offset b in the decision function is omitted [17] or integrated in the data space using homogenous coordinates [23, 24] as implemented in the LIBLINEAR library [25]. Here, the solution algorithms iterate over single samples and update the classification function parameters w and b of the decision function $f(x) = \operatorname{sgn}(\langle w, x\rangle + b)$ to the optimal values in relation to this sample. The *single iteration approach* creates a variant of a classification algorithm by performing this update only once. This directly results in *online learning*

algorithms. PAPs can be derived from C-SVM with this approach [7]. Online learning can be used to speed up the training procedure with low processing resources or to improve algorithms in terms of run time. In some cases it can deliver comparable performance to the original algorithm [18, 26, 27]. It is even possible to combine batch and online learning. First, the classifier is trained on a larger dataset with a batch learning. Then by using the *single iteration approach*, the connected online learning algorithm can be adapted and used in the application.

3 Optimization: pySPACE

There are several open source signal processing and machine learning libraries. Some important libraries are NumPy [28], SciPy [29], Modular Toolkit for Data Processing [30], Weka [31], LibSVM [22], and Scikit-learn [32]. pySPACE also provides a plethora of algorithms as depicted in Fig. 2 and wraps several libraries. For finding the best processing chain, access to numerous algorithms is helpful but only to a certain extend. In contrast to other libraries, pySPACE can be seen as a *high-level framework* which provides numerous methods for *both* classification *and* preprocessing. It automates the data processing, including loading and storing of the data, parallel processing of numerous different processing flows, and evaluation of the results. The interfacing to the data and algorithms is based on configuration files and *not* on scripts. The folder which contains all datasets which shall be processed, the parameters and algorithms which should be varied, and the list of algorithms which should be applied sequentially on the data, are the only user-defined specifications. Hence, our configuration files allow scientists with little programming experience to use the software. The streamlined format of a data processing configuration can be easily shared and compared even in publications. The present approach simplifies communication between scientists and would not be possible in the same way with scripts or graphical user interfaces (GUIs). The evaluation can be performed on a cluster for fast processing and provides a result tabular with numerous metrics [33] to analyze the differences between the compared algorithms and parameterizations. pySPACE was originally developed to allow for automatic benchmarking and tuning of EEG data processing chains [34–36]. A summary on respective evaluations is given in [9] as well as more details on pySPACE.

4 Decoding: Backtransformation

To understand classifiers, it is not only important to know the relations between them, but also to interpret them when they are applied on data. This understanding might lead to an improved processing chain or even to additional information about the data or the process which generated the data. Hence, new expert knowledge could be generated. A straightforward approach is to visualize the weights of the linear classification function [37, 38]. An extension to nonlinear classifiers has been suggested in [39] based on the derivative of the classification

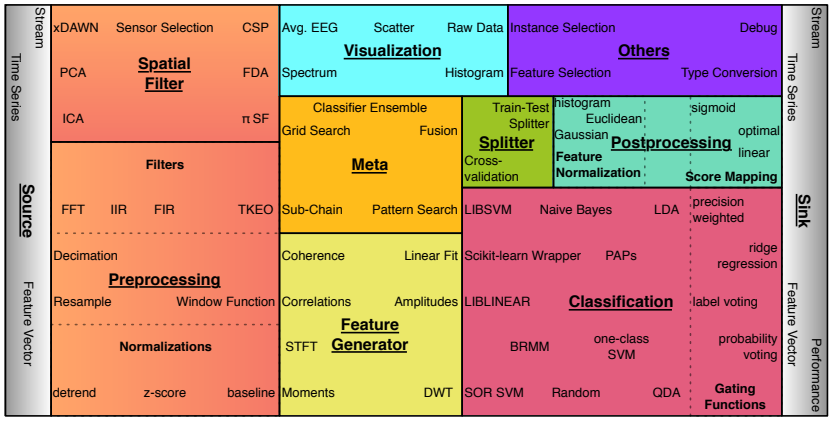
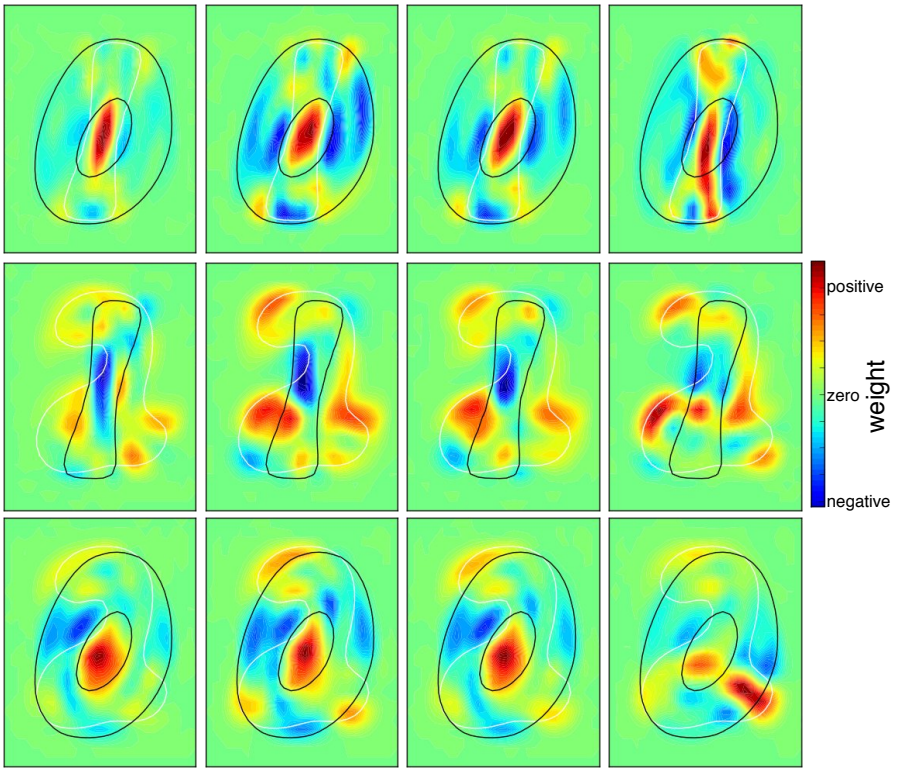


Fig. 2: Overview of the more than 100 processing nodes in pySPACE [9]. They are arranged according to processing categories (package names) and subcategories. The size of the boxes indicates the respective number of currently available algorithms.

function. Unfortunately, a derivative has to be calculated for every input sample which complicates the application and interpretation. The here proposed back-transformation concept is the extension of these methods to a complete signal processing chain, which ends with a (linear) classification function [40]. Therefore, the respective weights are calculated iteratively beginning with the classifier and going back through the processing chain. The final weights have the same format as the input data and could be visualized in the same way. Backtransformation is especially attractive when a dimensionality reduction algorithm is applied in the signal processing chain. In this case, an interpretation of the pure classifier weights is not informative without the weights of the dimensionality reduction algorithm. For nonlinear algorithms, a general transformation cannot be given anymore but it is possible to apply the chain rule and calculate the derivative of the signal processing function in the sample of interest. So for each input sample, a weight vector is obtained describing the *local* importance of each data component. Additionally to the visualization of the processing chain, back-transformation can be used to select features, to adapt a classifier to changing preprocessing (co-adaptation), and to enable sparse classification related to the input data (e.g., relevant time of the observed signal [35] or number of used sensors [34]).

5 Application Example

As a proof of concept, a classification on handwritten digit data was conducted (MNIST [41]). The classification of the digits 0, 1, and 2 is compared. First, the data was reduced in dimensionality with a principal component analysis



(a) RFLD (b) SVM (c) PAP (d) unary SVM

Fig. 3: *Contour plots of backtransformation weights for handwritten digit classification with different classifiers:* The white and black silhouettes display an average contour of the original data (digits 0, 1, and 2). The colored contour plots show the respective weights in the classification process. Negative weights (blue) are important for the classification of the first class (black silhouette) and positive weights (red) for the second class (white silhouette). Green weights are close to zero and do not contribute to the classification process. For the unary classification, the second class (white) was used.

(PCA) [42] from 784 to 40 and then normalized with a standardization (zero mean and variance of one on the given training data). For classification, a squared loss penalization of misclassifications was used to obtain a Gaussian loss in RFLD. RFLD, SVM, the respective SVM perceptron, and νoc-SVM were compared. Backtransformation can summarize all three processing steps and provides the respective weights belonging to the input data. This is visualized in Fig. 3. The classifiers itself do only determine the 40 weights of the normalized principal components. These weights would be difficult to interpret, but with the given backtransformation the weighting and its correspondence to the average

shapes can be observed. As expected due to the model similarities (single iteration approach) similar weight distributions were obtained for SVM and its online learning variant (PAP). The visualizations of SVM and RFLD look similar due to the connection with BRMM. However, for the distinction between the two digits 0 and 2 some larger differences can be observed. The one class classifier is different to the other classifiers as expected because it has been trained on a single digit only. Hence, characteristics of the other class can be only marginally observed due to the use of PCA which has been trained on both classes. This can be seen in the second and third row: although trained on the digit 2 in both cases, the classification results look different.

6 Conclusion

Optimizing the classification of spatiotemporal data is a difficult task which often requires expert knowledge. To ease this process especially for non-experts, three approaches are shown in this paper to improve the design and understanding of signal processing and classification with SVM variants. The pySPACE framework was presented, to process the data, tune algorithms and their parameters, and to enable the communication between scientists. Several connections between existing SVM variants have been shown and resulted in additional new SVM variants for unary classification and online learning. Due to the connections, it is easier to understand differences and similarities between the classifier variants and save time when implementing the classifiers. To finally interpret the complete signal processing chain which ends with a classifier, the backtransformation approach was presented. In case of solely affine transformations, it results in a representation of the processing chain, giving weights for each component in the input domain, which can be directly visualized. All three approaches were combined in an application on handwritten digit classification.

In future, the backtransformation concept should be implemented and tested on nonlinear signal processing chains. All three introduced concepts should be analyzed in further applications to prove their usefulness. The longterm goal is to make pySPACE a tool for autonomous learning.

References

1. Domingos, P.: A few useful things to know about machine learning. Communications of the ACM **55**(10) (2012) 78–87
2. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press (2000)
3. Müller, K.R., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B.: An introduction to kernel-based learning algorithms. IEEE Transactions on Neural Networks **12**(2) (2001) 181–201
4. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)
5. Vapnik, V., Others: The nature of statistical learning theory. 1995. Springer **120**(6) (2000)

6. Shivaswamy, P.K., Jebara, T.: Maximum relative margin and data-dependent regularization. Journal of Machine Learning Research **11** (2010) 747–788
7. Krell, M.M., Feess, D., Straube, S.: Balanced Relative Margin Machine The missing piece between FDA and SVM classification. Pattern Recognition Letters **41** (2014) 43–52
8. Mika, S.: Kernel Fisher Discriminants. PhD thesis, Technische Universität Berlin (2003)
9. Krell, M.M., Straube, S., Seeland, A., Wöhrle, H., Teiwes, J., Metzen, J.H., Kirchner, E.A., Kirchner, F.: pySPACE a signal processing and classification environment in Python. Frontiers in Neuroinformatics **7**(40) (2013) https://github.com/pyspace.
10. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing **14**(3) (2004) 199–222
11. Van Gestel, T., Suykens, J.A.K., Lanckriet, G., Lambrechts, A., De Moor, B., Vandewalle, J.: Bayesian framework for least-squares support vector machine classifiers, gaussian processes, and kernel Fisher discriminant analysis. Neural Computation **14**(5) (2002) 1115–1147
12. Schölkopf, B., Smola, A.J., Williamson, R.C., Bartlett, P.L.: New Support Vector Algorithms. Neural Computation **12**(5) (2000) 1207–1245
13. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Computation **13**(7) (2001) 1443–1471
14. Tax, D.M.J., Duin, R.P.: Support Vector Data Description. Machine Learning **54**(1) (2004) 45–66
15. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online Passive-Aggressive Algorithms. Journal of Machine Learning Research **7** (2006) 551 – 585
16. Eitrich, T., Lang, B.: Efficient optimization of support vector machine learning parameters for unbalanced datasets. Journal of Computational and Applied Mathematics **196**(2) (2006) 425–436
17. Steinwart, I., Hush, D., Scovel, C.: Training SVMs without offset. Journal of Machine Learning Research **12** (2009) 141–202
18. Krell, M.M., Kirchner, E.A., Wöhrle, H.: P300 Detection as an Unary Classification Problem. (submitted 2014)
19. Aggarwal, C.C.: Outlier Analysis. Springer New York, New York (2013)
20. Krell, M.M., Wöhrle, H.: New One-Class Classifiers based on the Zero Separation Approach. (submitted 2014)
21. Platt, J.C.: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In Schölkopf, B., Burges, C.J.C., Smola, A.J., eds.: Advances in Kernel Methods. MIT Press (1999) 61–74
22. Chang, C.C., Lin, C.J.: LIBSVM. ACM Transactions on Intelligent Systems and Technology **2**(3) (2011) 1–27
23. Mangasarian, O.L., Musicant, D.R.: Successive Overrelaxation for Support Vector Machines. IEEE Transactions on Neural Networks **10** (1998) 1032 – 1037
24. Hsieh, C.J., Chang, K.W., Lin, C.J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the 25th International Conference on Machine learning (ICML 2008), ACM Press (2008) 408–415
25. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A Library for Large Linear Classification. The Journal of Machine Learning Research **9** (2008) 1871–1874

26. Wöhrle, H., Krell, M.M., Straube, S., Kirchner, E.A., Kirchner, F.: An Online Adaptable Spatial Filter for User-Independent Single Trial Detection of Event-Related Potentials. (submitted 2014)
27. Wöhrle, H., Teiwes, J., Krell, M.M., Kirchner, E.A., Kirchner, F.: A dataflow-based mobile brain reading system on chip with supervised online calibration. In: Proc. International Congress on Neurotechnology, Electronics and Informatics, Vilamoura, ScitePress (2013) 46–53
28. Dubois, P.F.: Extending Python with Fortran. Computing Science and Engineering **1**(5) (1999) 66–73
29. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001) http://www.scipy.org/.
30. Zito, T., Wilbert, N., Wiskott, L., Berkes, P.: Modular toolkit for Data Processing (MDP): a Python data processing framework. Frontiers in Neuroinformatics **2**(8) (2008)
31. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software. ACM SIGKDD Explorations Newsletter **11**(1) (2009) 10–18
32. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research **12** (2011) 2825–2830
33. Straube, S., Krell, M.M.: How to evaluate an agent's behaviour to infrequent events? – Reliable performance estimation insensitive to class distribution. Frontiers in Computational Neuroscience **8**(43) (2014)
34. Feess, D., Krell, M.M., Metzen, J.H.: Comparison of sensor selection mechanisms for an ERP-based brain-computer interface. PLoS ONE **8**(7) (2013) e67543
35. Kirchner, E.A., Kim, S.K., Straube, S., Seeland, A., Wöhrle, H., Krell, M.M., Tabie, M., Fahle, M.: On the applicability of brain reading for predictive human-machine interfaces in robotics. PLoS ONE **8**(12) (2013) e81732
36. Krell, M.M., Tabie, M., Wöhrle, H., Kirchner, E.A.: Memory and Processing Efficient Formula for Moving Variance Calculation in EEG and EMG Signal Processing. In: Proc. International Congress on Neurotechnology, Electronics and Informatics, Vilamoura, ScitePress (2013) 41–45
37. LaConte, S., Strother, S., Cherkassky, V., Anderson, J., Hu, X.: Support vector machines for temporal classification of block design fMRI data. NeuroImage **26**(2) (2005) 317–329
38. Blankertz, B., Lemm, S., Treder, M., Haufe, S., Müller, K.R.: Single-Trial Analysis and Classification of ERP Components–a Tutorial. NeuroImage **56**(2) (2011) 814–825
39. Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., Müller, K.R.: How to Explain Individual Classification Decisions. Journal of Machine Learning Research **11** (2010) 1803–1831
40. Krell, M.M., Straube, S.: Backtransformation: A new representation of data processing chains with a scalar decision function. (under revision 2014)
41. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
42. Abdi, H., Williams, L.J.: Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics **2**(4) (2010) 433–459