A Fast Feature Selection Method for High-Dimensional MEG BCI Data

M. Spüler[1], W. Rosenstiel[1], M. Bogdan[1,2]

[1]Wilhelm-Schickard-Institute for Computer Science, University of Tübingen, Sand 13, 72076 Tübingen, Germany
[2]Computer Engineering, University of Leipzig, Johannisgasse 26, 04103 Leipzig, Germany

spueler@informatik.uni-tuebingen.de

Abstract

Magnetoencephalography (MEG) is a rarely used technique for BCI, which benefits are good signal quality and high spatial resolution. The latter can be seen as a drawback, when it comes to selecting the most important features or sensors. To increase accuracy and reduce classification time feature selection is an important step in signal classification. Due to the large amount of MEG sensors (≥ 275 possible) feature selection is very time consuming, which is why we present a fast feature selection method for high-dimensional data. It gives similar results as established feature selection methods and we show that its time-complexity grows linearly with the number of samples and is $O(n \log n)$ for the number of features.

1 Introduction

A Brain-Computer Interface (BCI) enables a user to communicate or control a computer by brain activity only. Brain activity can be measured by different methods like EEG or MEG. Using MEG has the benefit of recording from more sensors, which also results in having more dimensions in the feature space. Depending on the method of feature extraction being used, there can be an additional growth of features. Considering a MEG with $n = 275$ channels a spectral estimation by an autoregressive model in the range of 7–21 Hz in 2 Hz bins would result in $n \cdot 7 = 1925$ features. Using connectivity measures like coherence or phase synchronisation [1] results in $\frac{n\cdot(n-1)}{2} = 37675$ features. When dealing with such an amount of features it is important to have a method for feature selection, which reduces the full set of features to a small set including only the features that are most important for classification [2]. Another demand results from the limited computation power and time available. In most BCI experiments data is recorded in a first session and used for training a classifier, which is tested in an online-session some minutes later. In this case we need a fast feature selection method that can deal with a large number of features very quickly.

2 Methods

2.1 Data

For analysis and comparison of different feature selection methods we used the dataset from [3]. It consists of MEG data measured from 10 subjects. Each subject participated in 2 sessions and performed 7 different imagination tasks without feedback (51 trials per task and session). Since it was shown in [3] that imagined right hand movement and subtraction were the two best tasks to classify, we focused on data from these two tasks for evaluation. For feature extraction we used spectral coherence [1] resulting in 37675 features for all 275 channels.

2.2 r^2 Ranking

In a BCI context r^2 values are often used for performance estimation [4]. They can be seen as measure of correlation between a feature and the class membership. In other words: they give the proportion of variance for a feature, that is explained by the class membership.

Assume our data consists of n trials $x_i, i = 1, ..., n$ with each trial having m features $f_{ij}, j = 1, ..., m$ and a class label $y_i \in \{1, -1\}$. If n_1 trials are in class 1 and n_{-1} trials are in class -1, the r^2 value for feature j is calculated by:

$$r^2(j) = (\frac{(\sum_{i,y_i=1} f_{ij})^2}{n_1} + \frac{(\sum_{i,y_i=-1} f_{ij})^2}{n_{-1}} - \frac{(\sum_{i=1}^{n} f_{ij})^2}{n}) \cdot (\sum_{i,y_i=1} f_{ij}^2 + \sum_{i,y_i=-1} f_{ij}^2 - \frac{(\sum_{i=1}^{n} f_{ij})^2}{n})^{-1}$$

The result is a value between 0 and 1 where 0 stands for no correlation and 1 for perfect correlation (although perfect class seperability can be reached with $r^2(j) < 1$). When using r^2 values for feature selection, the r^2 values for all features are calculated, sorted in descending order and the features with the highest r^2 values are chosen for classification.

2.2.1 Incremental, Decremental Update of r^2 Values

R^2 values allow the possibility for incremental or decremental update. Having calculated the r^2 values for n samples, the solution for $n+1$ or $n-1$ samples can be calculated in constant time. By keeping in memory the following 6 intermediate results $(\sum_{i,y_i=c} f_{ij})^2$, $\sum_{i,y_i=c} f_{ij}^2$ and n_c for $c \in \{1, -1\}$ each incremental and decremental step can be calculated in constant time (20 numerical operations) by updating these intermediate results. Therefore the time complexity for calculating r^2 values grows linearly with the number of samples. Since the r^2 value for one feature is independent from all other features, each r^2 value can be calculated separately and the time complexity is also linear in respect to the number of features. Due to the ranking, where the r^2 values are sorted in descending order, the total time complexity for r^2-ranking is $O(n \log n)$.

The possibility of incremental and decremental update can be used for faster implementation of feature selection in cross-validations (CVs) or leave-one-out estimations (LOOEs). To ensure valid results a feature selection has to be performed on every training set each fold, which means performing n feature selections on a n-fold CV. By using decremental update it is possible to calculate the r^2 values with all trials only once and store the intermediate results in memory. In each fold the decremental update is used to remove the test trials from the full solution. The result is the same as the r^2 values would be calculated from the training set, but especially for LOOEs this is a big speedup, since the feature selection only needs to be done once and the results for each fold only need to be updated decrementally.

2.3 Performance Evaluation

To evaluate performance of r^2-ranking we tested some other feature selection methods: Recursive Feature Elimination (RFE) [5], Particle Swarm Optimization (PSO) [6], Principal Component Analysis (PCA) and Fast Correlation-Based Filter (FCBF) [7].

RFE is an iterative method that can be used with a Support Vector Machine (SVM). In each iteration a SVM is trained, a ranking criterion (feature weights in the linear case) is calculated and the feature with the smallest ranking criterion is removed. The RFE continues until the designated number of features is reached. To speed up the process we also used a modified version of the RFE in this paper (called mRFE later), which removes the 25 features with the lowest ranking criterion if the current number of features is greater than the designated number of features plus 100. After that only one feature is removed each iteration.

To compare computation time we used an Intel Dual Core E2180 at 2 GHz with 4 GB RAM running 64-bit Linux and 64-bit Matlab. Apart from LibSVM [8] only Matlab functions were used. We tested different numbers of starting features ranging from 250 to 37675 and selected the 100

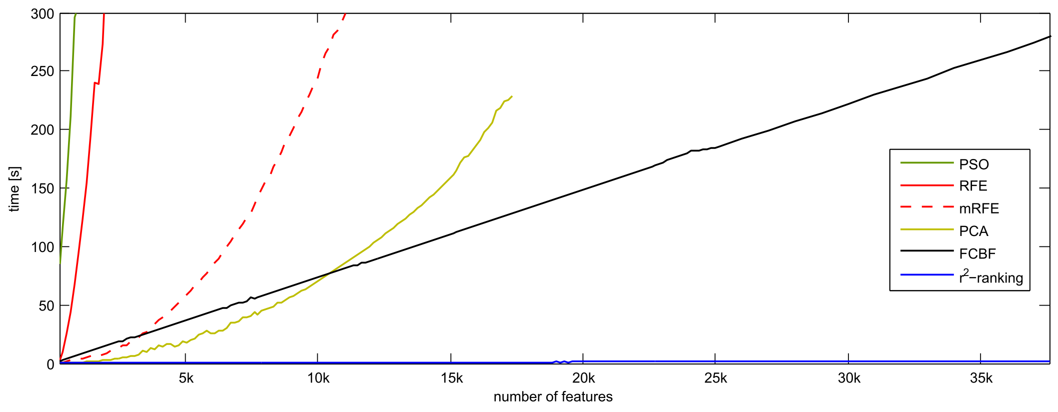

Figure 1: Averaged computation time to select the 100 best features for r^2-ranking, FCBF, mRFE, RFE, PCA and PSO with 204 trials. PCA could not be tested with more than 17350 features because the machine that was used for evaluation ran out of memory.

	optimal number of features			100 features	
method	# features	accuracy	time	accuracy	time
no feature selection	16110	80.2 %	-	-	-
FCBF	73	83.9 %	127 s	-	-
RFE	550	87.6 %	6986 s	83.9 %	7019 s
mRFE	550	87.5 %	608 s	84.4 %	261 s
r^2-ranking	330	86.5 %	0.79 s	83.8 %	0.79 s
PCA	50	83.1 %	146 s	78.5 %	146 s
PSO	9565	83.1 %	35758 s	-	-

Table 1: Comparison of different feature selection methods: number of features, accuracy and computation time needed for feature selection.

best features. For each method and each number of features we performed 10 runs with 204 trials and averaged the computation time.

To compare classification accuracies we used the first session of the data for feature selection and training and the second session to test the classifier. For classification we used LibSVM with a RBF-kernel. To test accuracy we excluded the outer MEG-sensors from the data, since they are supposed to have very little class related information but instead being more contaminated by artefacts than the inner sensors. This reduces the number of features to 16110.

3 Results

The results for computation time can be seen in Figure 1. It shows that r^2-ranking is faster than the other methods, especially for very high-dimensional data. Due to memory limitations it was not possible to calculate PCA with more than 17350 features on our machine.

The classification accuracy, number of features and computation time averaged over 10 subjects can be seen in Table 1 for the optimal number of features as well as a fixed number of 100 features. Since FCBF and PSO do not reduce to a specified amount of features, there are no results with 100 features.

While PCA, RFE and r^2-ranking show significantly better accuracy than without feature selection ($p < 0.01$, paired t-test), FCBF and PSO are not significantly better ($p > 0.1$). In addition there is no significant difference between r^2-ranking and RFE ($p > 0.25$).

To test the benefit of the decremental update we performed a LOOE with 204 trials and 37675 features. While the r^2-ranking without decremental update took 160.2 seconds, with decremental update it finished in 2.8 seconds. For comparison we also tried a 10-fold CV without decremental update that took 9.5 seconds. All times reported here only relate to the computation time needed for the feature selection, not for the whole LOOE or CV.

4 Discussion

The results show that r^2-ranking is faster than the other tested methods. In terms of accuracy there is no significant difference between r^2-ranking and RFE, but r^2-ranking yields significantly better results than without feature selection, thus making it a viable option for feature selection in high-dimensional data.

Another advantage of r^2-ranking is the possibility for incremental and decremental updates. Feature selection in CVs or LOOEs usually is the most time consuming step in the whole process. While the decremental update has proven useful to further speedup CVs or LOOEs, the incremental update could be used for an online feature selection, that updates the feature ranking every time a new trial is available without the need to know the previous trials. While PCA already has shown memory problems, the same problem arises for RFE if the number of trials becomes larger. Although the memory requirements for r^2-ranking are lower in general, the possibility for incremental update could help to further aid this problem.

5 Conclusion

We showed on high-dimensional data that r^2-ranking is superior to other feature selection methods concerning computation time. In terms of accuracy there is no significant difference to established feature selection methods like RFE. The possibility of incremental and decremental update also shows other interesting areas for application, like faster cross-validation or online feature selection.

References

[1] E. Gysels and P. Celka. Phase synchronization for the recognition of mental tasks in a brain-computer interface. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 12(4):406 –415, 12 2004.

[2] M. Bensch, M. Bogdan, and W. Rosenstiel. Phase synchronization in MEG for brain-computer interfaces. In *In Proceedings of the 3rd Int. Brain-Computer Interface Workshop*, pages 18–19, Graz, 09 2006.

[3] M. Bensch, J. Mellinger, M. Bogdan, and W. Rosenstiel. A multiclass BCI using MEG. In *Proceedings of the 4th Int. Brain-Computer Interface Workshop*, pages 191–196, Graz, 09 2008.

[4] H. Sheikh, D. J. McFarland, W. A. Sarnacki, and J. R. Wolpaw. Electroencephalographic (EEG)-based communication: EEG control versus system performance in humans. *Neuroscience Letters*, 345(2):89 – 92, 2003.

[5] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[6] C. J. Tu, L. Y. Chuang, J. Y. Chang, and C. H. Yang. Feature Selection using PSO-SVM. *IAENG International Journal of Computer Science*, 33:111–116, 2007.

[7] L. Yu and H. Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *Proc. 20th Int'l Conf.Machine Learning*, pages 856–863, 2003.

[8] C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.