Discovering Patterns in EEG-Signals: Comparative Study of a Few Methods

Miroslav Kubat[1], Doris Flotzinger[2], and Gert Pfurtscheller[1]

[1] Ludwig-Boltzmann Institute of Medical Informatics and Neuroinformatics
Graz University of Technology, Brockmanngasse 41, 8010 Graz
e-mail mirek@fbmtds04.tu-graz.ac.at
[2] Department of Medical Informatics, Institute of Biomedical Engineering
Graz University of Technology, Brockmanngasse 41, 8010 Graz

Abstract. The objective of this paper is to draw the attention of the *ML*-researchers to the domain of data analysis. The issue is illustrated by an attractive case study—automatic classification of non-averaged *EEG*-signals. We applied several approaches and obtained best results from a combination of an *ID3*-like program with Bayesian learning.

1 Introduction

The general task of machine learning applied to data analysis is to facilitate the classification of unseen examples, to extract relevant information from the data, and to provide intelligent interpretation of the classification scheme.

The research reported here relates to a broader project, aiming at the development of a Brain-Computer Interface (*BCI*), a direct link between the brain and an electronic device. The objective is to discover typical patterns in the measured *EEG*-signals in order to recognise simple commands such as 'move right,' 'move left,' 'move up,' or 'move down.' A description of the *BCI* with first results are published in (Pfurtscheller, Flotzinger & Kalcher, 1992).

Each example submitted to the learner represents a single *EEG*-measurement consisting of 70 numerical values, and is classified so that each class stands for one command. We started with the simple task of discerning 'move left' from 'move right.' Detailed discussion of the data acquisition and preprocessing would go beyond the scope of this paper. Suffice it to say that, for each learning example, potentials were measured on 14 electrodes placed on an intact scalp of a test person, in the frequency band of 8-12 Hz in 5 time-slices preceding the movement of the left or right hand. Thus we had 70 real-valued attributes and two classification classes.

We applied several techniques to the analysis of three sets of examples. Each of these sets was obtained from a different test person. The sizes of the sets were 213, 140, and 246 examples, respectively. For each person, we used 60% of the examples for learning and the rest for testing. All results were averaged over 10 different random selections of learning examples. (In the *BCI*-project, the machine is always trained for one particular person.)

Two phenomena characterized the data: they were sparse and of low quality. *Sparseness* relates to the vast number of all possible combinations of attribute values, from which only tens or hundreds are available as learning examples, without any guarantee of representativeness. The *low quality* of the data is caused by the different degree of attention of the tested persons, by changes in vigilance, by artefacts during recording, by not using the optimal frequency band, and by other factors. The examples are noisy, and perhaps do not contain the expected information altogether, because waves measured on a single frequency band of 14 electrodes are certainly insufficient for 'reading thoughts.' Also, there is a high probability that most of the attributes are irrelevant.

In the search for the best suited data-analysis method, we postulated three minimum requirements:

1) The inferred pattern must be maximally *accurate* so as to enable sufficiently precise classifications;
2) The results must be as *stable* as possible—i.e. the classification accuracy must be about the same on each of the data sets;
3) The method should enable *interpretation*, so that the researchers can use the results for further improvement of the experimental setting.

2 Brief Outline of the Applied Methods

We have applied two subsymbolic methods (Multilayer Perceptron and Learning Vector Quantizer), and three symbolic methods (Bayesian Classifier, Induction of Decision Trees, and a combination of these two).

Multi-Layer Perceptron (*MLP*)

The *MLP* is a traditional neural network whose topology builds on a few layers of processing units—an *input* layer consisting of d nodes, where d is the number of attributes or the dimension of the input vector (in our case $d = 70$); a *hidden* layer whose size is user-defined; and an *output* layer consisting of c nodes, where c is the number of classes (two, in our case). Inputs to the individual units are weighted. The output of each unit (except for the input layer) is a function of the weighted sum of its inputs: we applied the common sigmoid *activation* function. For more detailes, see (Rumelhart, Hinton, & Williams, 1989).

In the *classification* phase, the values of each individual attribute are propagated through the network. The class assigned to the output node with the highest value of the activation function is used as the classification value.

Learning is carried out by error propagation from the output to the input layer. The difference between desired and actual output of the network is used to correct the weights at each layer. Usually a predetermined number of runs through the training set is performed. The initial weights are set to small random values.

We experimented with several topologies of the *MLP* and achieved the best results for one hidden layer with ten units.

Learning Vector Quantizer (*LVQ*)

The essence of the *LVQ* (Kohonen, 1990) is in dividing the d-dimensional space into a predefined number of regions. Each region is represented by a so called *reproduction* vector which is assigned a classification value.

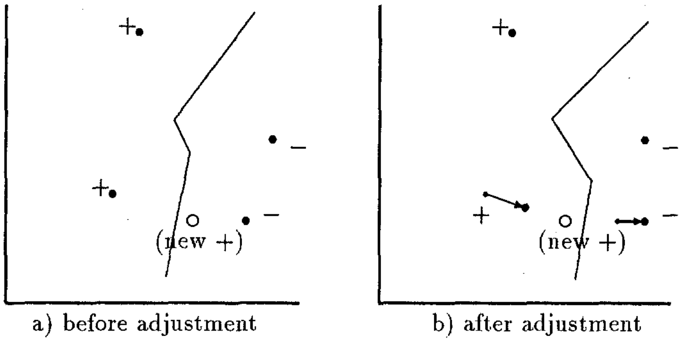

Fig. 1. *LVQ:* two closest reproduction vectors adapt to the new arrival.

In the *classification* phase, the new example is assigned the classification value of the closest reproduction vector.

The *learning* scheme consists of two parts: (1) *initialization* defines the initial boundaries of the regions and the respective reproduction vectors; (2) *tuning* is a trial-and-error mechanism: the system takes the examples one by one and tries to classify them. If the classification attempt is wrong, the reproduction vector is pushed slightly away (in the d-dimensional learning space) from the example, and the second closest reproduction vector, if it is of the same class as the example, is pulled closer to the example. These adjustments are performed for a predetermined number of runs through the whole training set.

For the initialization, we followed the recommendation of (Peltoranta, 1992) to use k-means. The parameter k determines the ideal number of reproduction vectors. The best results were achieved for $k = 2$ and 3.

Conversion to Symbolic Values

Any transformation from numeric to symbolic values entails information loss. However, it can significantly reduce the learning space and stress 'regularities' in the data. The conversion algorithm we have implemented is based on entropy minimization. For each attribute, the following procedure was applied:

1) Find the minimum and maximum value of the attribute. Compute the entropy of the whole interval by $H_{initial} = \Sigma_{i=1}^{c} p_i \log p_i$, where c is the number of classes (here, $c = 2$), and p_i is the probability of the i-th class;
2) Find the optimal split of the interval into subintervals minimizing the overall entropy $H = H_1 + H_2$, where H_1 and H_2 are entropies of the individual intervals;

3) If the difference between the previous entropy and the new entropy is less then p_{min} percent, then *stop*. Otherwise, pick the interval with the highest entropy and find its ideal split, maximizing the overall entropy;
4) Repeat the previous step until either the algorithm fails to improve the overall entropy by at least p_{min} percent or the number of intervals has reached the user-specified maximum.

Typical number of intervals per attribute was 4. Typical value for p_{min} was 5%.

Bayesian Classifier *B*

This simple approach gives surprisingly good results even if the general requirement of the pairwise independency among the attributes is not satisfied.

In the *classification* phase, the example is assigned the class C_i for which the following formula has the maximum value:

$$P(C_i \mid v_1, \ldots, v_d) = P(C_i) \cdot \Pi_{k=1}^{d} P(v_k \mid C_i)$$

where d is the number of attributes, $v_1, \ldots, v_d$ are the attribute values of the example to be classified, $P(C_i)$ is the a priori probability of the occurence of class C_i, and $P(v_k \mid C_i)$ is the a priori probability of the k-th attribute value in class C_i. The probabilities are calculated as relative frequencies in the data (even though they may not be well statistically grounded for rare values v_i).

Induction of Decision Trees *IDT*

The *learning* principle of the algorithm for the induction of decision trees consists of growing the tree and pruning the tree. Growing is carried out by the following procedure:

Find the attribute with the maximal information content and place it at the root of the tree. The n distinct values of the attribute divide the data set into n subsets. If all examples of a subset belong to a single class, then make the subset a leaf assigned a label of this class. Otherwise, find in each of the subsets the best attribute, splitting the subset into 'subsubsets,' and so on until either all of the remaining subsets are empty or assigned a label, or until there is no unused attribute left (in the last case, the final subset will be assigned more than one class). For a formula deciding which attribute is best, see (Quinlan, 1986).

When the tree is constructed, the next step is its pruning to avoid overspecialized descriptions and to discard noise (Niblett, 1987). Pruning consists in cutting off those branches that are not well statistically grounded.

The *classification* procedure consists in propagating the example through the tree starting from the root and testing at each node the respective attribute value. Depending on its value, the branch leading to the next node is selected. When a leaf is reached, then its class is assigned to the example.

IDT initializing Bayes

IDT is used to reduce the dimension of the learning space which is then analyzed with the Bayesian Classifier. The procedure consists of the following three steps:

1) Run *IDT* on the d-dimensional data and build a decision tree;
2) Discard all attributes that have not appeared in the decision tree. The new dimension of the space is $d1 \leq d$;
3) Determine a priori probabilities (in the $d1$-dimensional space) to serve the Bayesian Classifier.

3 Experimental Results

The most general results in *accuracy* are summarized in Table 1. As expected, the Bayesian Classifier, if run on all 70 attributes, is the clear loser because it lacks any ability to discern between relevant and irrelevant information. The subsymbolic approaches scored better. However, they were not very stable on data files of different quality—*MLP* achieved the absolutely best score on C87B but was practically useless on C05B because 50% accuracy can be achieved by mere tossing a coin. Hence, the subsymbolic methods are more sensitive to the quality of the learning data. The favourable results of *IDT* can be explained by its ability to assess the relevance of the individual attributes.

Table 1. Accuracy of the predictions of the side of hand movement from *EEG* data recorded prior to the actual movement.

	C05B	C16B	C87B	Average	st.dev.
MLP	49.7	72.1	84.1	68.6	14.3
LVQ	51.6	71.6	82.2	68.5	12.7
Bs	60.7	59.3	70.9	63.6	5.2
IDT	71.3	73.6	79.3	74.7	3.4
IDT-Bs	70.5	77.3	79.9	75.9	3.9

The absolute winner is the method that extracts the most significant attributes by means of *IDT* and then runs Bayes on them. The reason is that the ordering imposed by the decision tree is too rigid. Instead of a precise ordering of the attribute tests, Bayes supplies conditional probabilities and thus allows for more flexibility. The results of this approach are also relatively *stable*.

Apart from accuracy and stability, the Introduction postulated also the important requirement of *interpretability*. Though it *is* possible to extract knowledge from Neural Networks (Towell, Craven & Shavlik, 1991), the process is far from being easy and straightforward. The results from *IDT*, in turn, can immediately be interpreted, for instance, 'most important is the potential on electrodes 5 and 10 in time slice 2, and then the potential on electrode 14 in time slice 4.' This helps the user to develop a deeper understanding of the results of the analysis and supply guidelines for further experiments.

4 Conclusion

For data analysis in vast learning spaces, the analyst must be acquainted with a rich repertoir of methods with their pros and cons, and be able to apply them in a flexible manner.

To deal only with numeric data is an oversimplification of a real-world task. In our future research, we want to enrich the original data so that they contain also results of measurements at other frequency bands and predicates such as 'activation on right hemisphere precedes activation on the frontal electrodes.' It is encouraging to know that the combination *IDT*–Bayes worked well because this will work even if Boolean variables are added. Also, the experience saying that symbolic analysis tends to produce better results than numerical is positive because symbolic analysis is usually much faster than numerical.

Acknowledgements

Thanks are due to Wolfgang Mohl for data recording and preprocessing. The research was sponsored by grants from 'Fonds zur Förderung der wissenschaftlichen Forschung' (Project *S*49/02 and Lisa-Meitner fellowship *M003 MED* for M.Kubat) and from the Austrian Federal Ministery of Science and Research ($Gz5.167/2-27b/91$)

References

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE,* Vol.78, No.9, pp. 1464–1480

Niblett, T. (1987). Constructing Decision Trees in Noisy Domains. In Bratko,I.–Lavrač,N. (eds.) *Progress in Machine Learning.* Sigma Press, Wilmslow

Peltoranta M. (1992). Methods for Classification of Non-Averaged *EEG* Responces using Autoregressive Model Based Features. PhD-thesis. Graz University of Technology.

Pfurtscheller, G., Flotzinger, D., and Kalcher, J. (1992). Brain-Computer Interface—A New Communication Device for Handicapped Persons. In: Zagler,W. (ed): *Computer for Handicapped Persons: Proceedings of the 3rd International Conference,* Vienna, 409–415

Quinlan, J.R. (1986). Induction of Decision Trees. In: *Machine Learning 1,* 81–106

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1989). Learning Internal Representations by Error Propagation. In: Rumelhart,D.E. et al. (eds.): *Parallel Distributed Processing,* MIT Press, Cambridge, MA, Vol.1, pp.318–362

Towell, G.G., Craven, M.W. and Shavlik.J. (1991). Constructive Induction in Knowledge-Based Neural Networks. *Proceedings of the 8th International Workshop on Machine Learning,* Morgan Kaufmann, San Mateo