Multiclass Disease Identification Employing Functional Protein Micorarrays

Patrick Kalmbach and David Suendermann-Oeft

***Abstract* — In this paper, we tackle a five-class problem for distinguishing Alzheimer's Disease, Parkinson's Disease, Breast Cancer, Multiple Sclerosis, and healthy control patients. For this purpose, we used the raw microarray data from a publicly available corpus. We tested performance of 25 base learners, investigated the influence of feature selection on classification performance, applied parameter tuning and boosted performance even further using ensemble learning. We found AdaBoostM1 with J48 (open-source implementation of C4.5) as base learner performing best, with a classification error rate of 8.8%.**

I. Introduction

Diseases such as Alzheimer's Disease, Parkinson's Disease, Multiple Sclerosis, and Breast Cancer affect large parts of the population. Early detection of these diseases remains a problem despite huge efforts and successes. Often, a disease is diagnosed based on visible symptoms only, at which point the patient may face substantial damage already. Early detection and treatment, however, can help to slow down the progress of the disease, help mitigate symptoms and improve the overall live quality of patients.

It has been shown, that autoantibodies in human blood are reliable biomarkers for identifying these diseases [1-3]. By using protein microarrays, the authors were able to identify small sets of antibodies differentially expressed for patients with one of the respective diseases. This knowledge can be used to develop new drugs and therapeutic methods.

The selected antibodies allowed for a very accurate identification of the respective disease. However, they were not the only antibodies differentially expressed. Restriction to this small subset, information contained in the remaining antibodies cannot be used. Furthermore, in [1-3] RandomForest [4] was used as sole classifier to establish the predictive accuracy of the selected antibodies.

In this paper, we tackle a five-class problem distinguishing Alzheimer's Disease, Parkinson's Disease, Multiple Sclerosis, Breast Cancer and healthy controls. We present results of different base learners, investigate the influence of feature selection on classification performance and improve classification performance further utilizing ensemble learning.

To allow the scientific community to reproduce the experiments described in this work, we chose a publicly available corpus and used the open-source software kit Weka[1], which had previously been used successfully in context of gene expression microarrays [5]. Required changes to handle microarray data with Weka were implemented in Java. This code is also made publicly available[2].

The remainder of this paper is structured as follows: In Section II, we briefly discuss the corpus and evaluation methodology used in our study. In section III, we report experimental results before drawing conclusions in Section IV.

II. Materials and Methods

A. Materials and Methods

The corpus used in this work was first described in a study by Han et alters in [1] and is publicly available on the MIAME-compliant NCBI GEO database [6] under the accession number GSE29654. The corpus consists of microarrays of 159 patients. The detailed composition of the corpus is given in Table I.

TABLE I. DETAILED DEMOGRAPHIC INFORMATION OF PATIENTS

Disease	*Samples*	*Age Mean*	*Age Range*	*Sex male %*
Parkinson's Disease	29	74.0	53 to 88	55
Alzheimer's Disease	50	78.5	61 to 97	40
Multiple Sclerosis	10	46.0	27 to 59	30
Breast Cancer	30	46.7	32 to 54	0
Older Controls	20	57.7	51 to 86	100
Younger Controls	20	24.7	19 to 30	65

The authors of [1] used ProtoArray v5.0 Human Protein Microarrays from Invitrogen to identify autoantibodies in human sera. Each microarray contained 9,486 unique human protein antigens to identify autoantibodies. Proteins are printed in duplicate providing 18,972 data points.

When analyzing the results with the freely available analysis software ProtoArray Prospector v5.2 [3] from Invitrogen, only half of the data points are used [7]. We decided to utilize all data points, therefore basically doubling the corpus to 318 instances.

To not miss any information, we did not apply any feature normalization method [8-9] as done in previous work. Instead, we used the raw signal derived from the difference of foreground and background intensity available from the aforementioned database.

Patrick Kalmbach is with Hewlett Packard, Boeblingen, Germany (email: Patrick.Kalmbach@live.com),

David Suendermann-Oeft is with ETS, San Francisco, USA. At the time this work was conducted, he was with DHBW Stuttgart, Germany, (e-mail: david@suendermann.com)

[1] http://www.cs.waikato.ac/nz/ml/weka

[2] http://suendermann.com/diseaseClassification

[3] http://www.lifetechnologies.com/de/home/life-science/protein-expression-and-analysis/biomarker-discovery/protoarray/resources/data-analysis.html

B. *Weka*

Weka [10] is a popular open-source toolbox for data mining offering a large variety of state-of-the-art classifiers.

As the corpus is rather small in size, we decided to run leave-one-(patient)-out cross-validation. In this technique, a model is trained on all instances except for those of a given patient. These instances are used for testing. This procedure is repeated for all patients, and the results are finally consolidated.

In our case, the training ser for each fold consists of 316 instances and the test set of two instances from one patient. This separation by patient is crucial, since the results would else be overoptimistic. Using this method no microarray data from the individuals is in the training and test set at the same time.

III. Experiments

A. *Default Options*

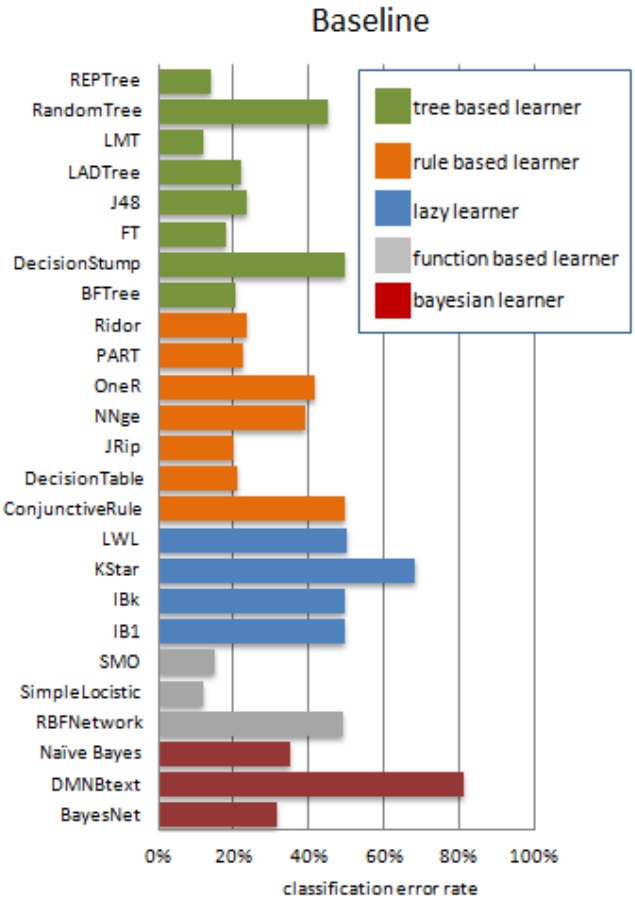

Figure 1. Baseline of different classification schemes available in Weka

Baseline First, we investigated how different classifiers behave on the problem at hand. To this end, we used 23 base learners provided in the Weka and applied leave-one-out cross validation, as described above. The results are depicted in Figure 1. All 9,486 autoantibodies were considered, and the default settings provided by Weka were used for each classifier.

We found that function- and tree-based learners performed best with a classification error rate of as low as 12%. Rule-based learners did not perform as well with the lowest classification error rate of 20%. Byesian and lazy learning schemes performed worst with classification error rates between 30% and 63%.

The poor results for lazy learning schemes, is somewhat unexpected as those schemes often provide god performance. Other popular learners like Bayes [11], Support Vector Machines [12], C4.5 (J48) [13], Logistic Regression [14] or JRip [15] perform well for this problem.

Feature Selection Inspired by previous work [1], we used attribute selection by information gain in order to investigate the influence of selecting different antibodies on the performance. We used a ranker discarding all features whose information gain is below a certain threshold. Results of these experiments are shown in Figure 2.

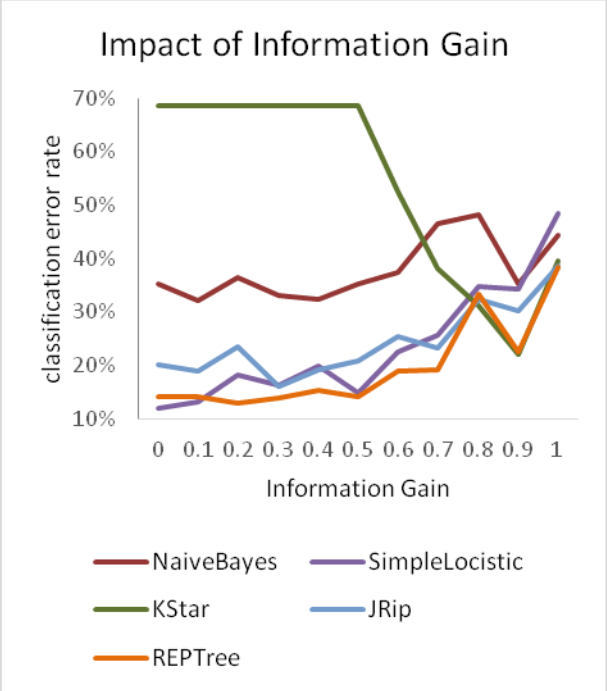

Figure 2. Dependency of classifier performance on feature selection by information gain of five exemplary base learners

The classification error of rule- and tree-based learners remains relatively unaffected by information gain between 0 and 0.5 after which the error rate start to increase. Function-based, lazy and Bayesian learners show a higher sensitivity towards feature selection. However, most of them have their minimal classification error between an information gain of 0 and 0.75.

These results are reasonable as rule- and tree-based learners of the use information gain themselves to decide when to split or create rules. The more features are taken away, however, the more this selection process is affected.

B. *Parameter Tuning*

SimpleLogistic SimpleLogistic is a classifier for building linear logistic regression models. In order to fit the logistic models, SimpleLogistic uses LogitBoost [16] with simple regression functions as base learners. SimpleLogistic produces the best baseline with a classification error of less than 12%. Using a grid search, we tuned the beta value for weight trimming (*W*) for LogitBoost and the parameter for early stopping of LogitBoost based on heuristics (*H*). The tested values are *W = 0, 0.1, …, 0.9* where zero means no weight trimming and *H `0, 25, …,* 100 where zero means no heuristic stop. The results are depicted in Figure 3. The number of LogitBoost iterations is cross-validated. However, by setting the parameter H to a value greater than zero, LogitBoost will terminate, if no new error minimum has been found in the last *H* iterations.

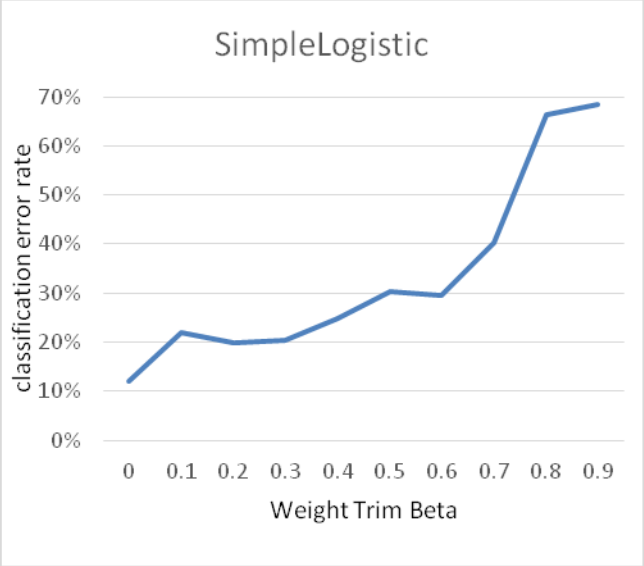

Figure 3. Classification Error Rate for SimpleLogistic depending on paramter W an H. As H has no influence on classification error rate, one line is visible only

We found that parameter *H* did not have any effect at all on the performance of SimpleLogistic. In contrast, using weight trimming has a strong negative impact on the classification error. Assumptions made in [14] hold not true for this specific problem. In general, weight trimming can greatly improve computation performance without loss of accuracy. This is achieved by omitting attributes with very small weight

C. *Ensemble Learning*

Bagging Bagging [17] is a popular ensemble learning scheme utilizing multiple versions of a single base learner combining them by majority vote [10]. The implementation of bagging we used in this work has two main parameters with potential impact on classification performance. The first, being the number of iterations (*I*) and the other the size of the bag in percent of the training set. We used again a grid search with I = 10, 20, …, 100 and bag size of P = 25, 50, …, 100. The results are depicted in Figure 4.

The minimal classification error with 11.94% was achieved with 100 iterations and a bag size of 50%.

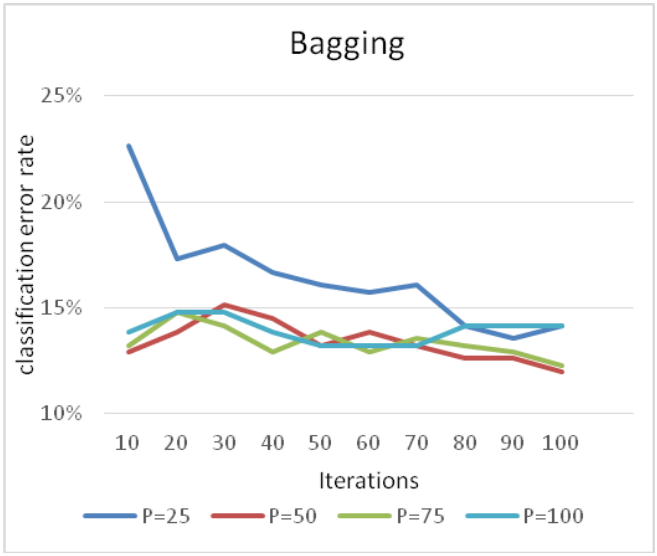

Figure 4. Classification error rate for boosting depending on the number of iterations *I* and bag size *P*

RandomForest In previous studies [1-3], excellent results were achieved using RandomForest [4]. RandomForest is also an ensemble learning scheme using RandomTree as base learner. In our experiment, we grew an unlimited tree and changed the number of attributes for the random selection *K = 0, 10, …, 100*. The results are depicted in Figure 5.

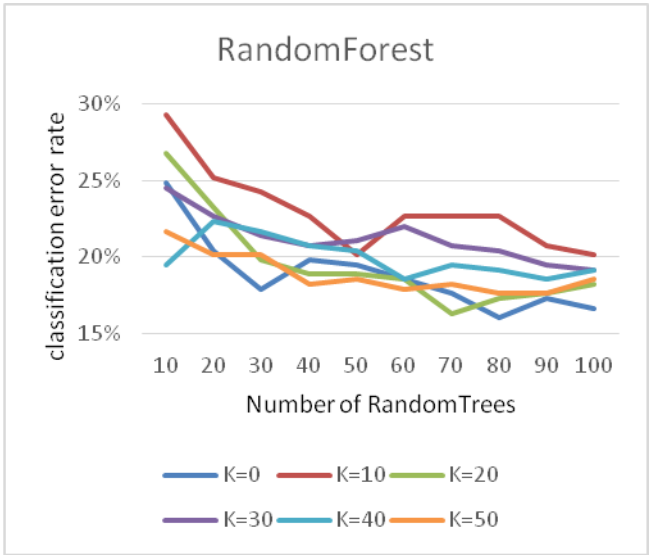

Figure 5. Classification error rate for RandomForest iterating over the number of RandomTrees *I* and number of features to consider in random selection *K*

The minimal classification error we achieved was 16.04% with *I = 80* and *K = 0*. RandomForest thus performed worse than bagging, boosting, and SimpleLogistic.

AdaBoostM1 AdaBoostM1 [18] is an ensemble learning scheme utilizing boosting. Being related to bagging, boosting concentrates on subsequent iterations of previously incorrectly classified instances [10].

To get started, we used the default configuration provided by Weka. Here, *DecisionStump* was used as base learner and

AdaBoostM1 run with ten iterations and weight pruning of 100. This combination resulted in a classification error of 50%.

Next we applied *J48* as base learner. *J48* is the open-source implementation of the simple tree-based learner *C4.5*. Therefore, *J48* is more likely to produce alternating results in presence of slight changes in the data set, as opposed to more complex schemes like *LMT*, being an essential property for applying *Boosting*.

In a first run we used *AdaBoostM1* with different values for weight threshold and number of iterations. We observed a gradual decrease in classification error with increasing threshold and number of iterations.

Going from this observation, we evaluated AdaBoostM1 with J48 with higher values for iterations and weight threshold. Figure 6 shows the learning curves for this setup. Using this approach, we could achieve a classification error of 8.81% which equals 28 wrongly classified instances or 14 patients.

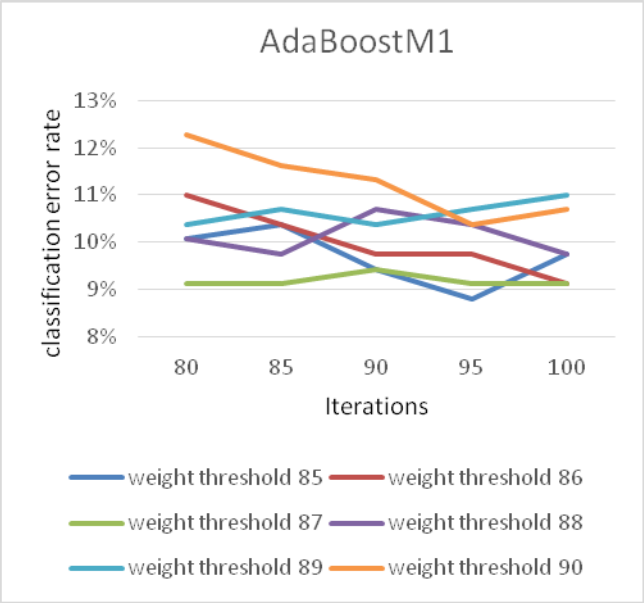

Figure 6. Learning curves for AdaBoostM1 with J48 as base learner

IV. Conclusion and Future Work

In this paper, we demonstrated how Alzheimer's Disease, Parkinson's Disease, Multiple Sclerosis, Breast Cancer and healthy controls can be distinguished with an accuracy of more than 91% using boosting and unprocessed data from functional protein microarrays. However, the corpus used in this study is relatively small. Therefore, in order to verify the presented findings, testing on a larger corpus would be preferable. In addition, other ensemble learning schemes such as *Stacking* could be evaluated in the future. Moreover, it would be interesting to see how feature normalization such as *proCAT* [8] or *Robust Linear Regression* [9] affects the outcome of this classification problem.

References

[1] M. Han, E. Nagele, C. DeMarshall, N. Acharya, and R. Nagele, "Diagnosis of Parkinson's disease based on disease-specific autoantibody profiles in human sera, " *PLoS ONE*, vol. 7, no. 2.

[2] B. Ayoglu, A. Hggmark, M. Khademi, T. Olsson, M. Uhln, J. M. Schwenk, and P. Nilsson, "Autoantibody profiling in multiple sclerosis using arrays of human protein fragments, " vol. 12, no. 9, pp. 2657–2672, 2013.

[3] K. S. Anderson, S. Sibani, G. Wallstrom, J. Qiu, E. A. Mendoza, J. Raphael, E. Hainsworth, W. R. Montor, J. Wong, J. G. Park, N. Lokko, T. Logvinenko, N. Ramachandran, A. K. Godwin, J. Marks, P. Engstrom, and J. LaBaer, "Protein microarray signature of autoantibody biomarkers for the early detection of breast cancer, " *Journal of Proteome Research*, vol. 10, no. 1, pp. 85–96, 2011.

[4] L. Breiman, "Random forests, " *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] J. E. Gewehr, M. Szugat, and R. Zimmer, „BioWeka–extending the Weka framework for bioinformatics," vol. 23, no. 5, pp. 651–653, 2007.

[6] R. Edgar, M. Domrachev, and A. E. Lash, "Gene expression omnibus: Ncbi gene expression and hybridization array data repository, " vol. 30, no. 1, pp. 207–210, 2002.

[7] Immune Response Biomarker Profiling Toolbox v5.2, Invitrogen, April 2009.

[8] X. Zhu, M. Gerstein, and M. Snyder, "Procat: a data analysis approach for protein microarrays, " *Genome Biology*, vol. 7, no. 11, p. R110, 2006.

[9] A. Sboner, A. Karpikov, G. Chen, M. Smith, M. Dawn, L. Freeman-Cook, B. Schweitzer, and M. B. Gerstein, "Robust-linear-model normalization to reduce technical variability in functional protein microarrays, " *Journal of Proteome Research*, vol. 8, no. 12, pp. 5451–5464, 2009.

[10] I. Witten, E. Frank, and M. Hall, Data mining: practical machine learning tools and techniques. Elsevier.

[11] M. Friedman and D. Geiger, "Bayesian network classifiers, " in *Machine Learning*.

[12] J. Platt, "Fast training of support vector machines using sequential minimal optimization, " in *Advances in Kernel Methods – Support Vector Learning*

[13] R. Quinlan, "C4.5: Programs for Machine Learning. " Morgan Kaufmann Publishers

[14] N. Landwehr, M. Hall, and E. Frank, „Logistic model trees."

[15] W. W. Cohen, "Fast effective rule induction, " in *Twelfth International Conference on Machine Learning*.

[16] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting, " Stanford University, Tech. Rep., 1998.

[17] L. Breiman, "Bagging predictors, " *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[18] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm, " in *Thirteenth International Conference on Machine Learning*, San Francisco, 1996, pp. 148–156