

Performance Prediction Challenge

Isabelle Guyon, Amir Reza Saffari Azar Alamdari, Gideon Dror, and Joachim M. Buhmann

Abstract—A major challenge for machine learning algorithms in real world applications is to predict their performance. We have approached this question by organizing a challenge in performance prediction for WCCI 2006. The class of problems addressed are classification problems encountered in pattern recognition (classification of images, speech recognition), medical diagnosis, marketing (customer categorization), text categorization (filtering of spam). Over 100 participants have been trying to build the best possible classifier from training data and guess their generalization error on a large unlabeled test set. The challenge scores indicate that cross-validation yields good results both for model selection and performance prediction. Alternative model selection strategies were also sometimes employed with success. The challenge web site keeps open for post-challenge submissions: <http://www.modelselect.inf.ethz.ch/>.

I. INTRODUCTION

In most real world situations, it is not sufficient to provide a good predictor, it is important to assess accurately how well this predictor will perform on new unseen data, i.e. predicting the “generalization” performance of the predictor. Before deploying a model in the field, one usually performs a “pilot study” to investigate whether one or several proposed models meet desired specifications or whether one should invest more time and resources to collect additional data and/or develop more sophisticated models. This tradeoff raises the problem of making best use of a limited amount of available data to both train the models and assess their performances. Since model selection is rendered trivial if model performances are known with precision, model selection and performance prediction are tightly related.

The problems of model selection and predicting generalization performance have long been studied in statistics and learning theory [1]. It is known that the training error is usually a poor predictor of generalization performance, unless the training set is very large compared to the capacity or complexity of the trained model.¹ This fact has motivated the development of various hold out techniques, including leave-one-out, k-fold cross-validation, and bootstrap (see e.g. [2] for a review). Another strategy has been to develop theoretical performance bounds, which correct the overly optimistic training error by adding a term taking into account

the ratio of the capacity of the model and the number of training examples [3].

Some empirical studies have been performed to attempt to determine the relative efficacy of the various methods [4], [5], [6]. The availability of large data repositories, including the well known UCI Machine Learning repository [7], and dozens of other sites [8], makes it possible to conduct systematic studies. Yet, no consensus seems to be emerging, possibly due to the bias each team may have introduced by promoting its favorite method. While no empirical study can be completely unbiased, competitions offer the advantage that all the methods under study are treated with the best possible care by competitors who are promoting them. Competitions provide a concentration of energy at a given point in time of many researchers working on the same topic, which pushes the frontiers of the state-of-the-art. Many conferences have recognized these benefits and organize competitions regularly (e.g. KDD, CAMDA, ICDAR, TREC, ICPR, and CASP). These considerations motivate our present effort for WCCI 2006, which includes a competition on performance prediction and a special session on model selection.

Our challenge design is inspired by previous competitions, and particularly by the challenge in feature selection that we co-organized for the NIPS03 workshops [9]. We formatted five data sets for the purpose of benchmarking performance prediction in a controlled manner. The data sets span a wide variety of domains and have sufficiently many test examples to obtain statistically significant results. The WCCI 2006 performance prediction challenge is to obtain a good predictor AND predict how well it will perform on a large test set.² The entrants had to submit results on ALL five data sets provided. To facilitate entering results for all five data sets, all tasks are two-class classification problems. During the development period, participants could submit results on a validation subset of the data to obtain immediate feedback. The final ranking was performed on a separate test set.

We estimated that 145 entrants participated. We received 4228 “development entries” (entries not counting towards the final ranking). A total of 28 participants competed for the final ranking by providing valid challenge entries (results on training, validation, and test sets for all five tasks). We received 117 submissions qualifying for the final ranking (a maximum of 5 entries per participant was allowed). The participation doubled in number of participants and entry volume compared to the feature selection challenge.

²It is trivial to predict the performance of a bad predictor performing a random assignment of patterns to labels, therefore we impose that the predictors must be good to win.

Isabelle Guyon is an independent consultant, Berkeley, California, USA, presently a visiting professor at ETH Zurich (isabelle@clopinnet.com).

Amir Reza Saffari Azar Alamdari is with the ICG, Graz University of Technology, Austria.

Gideon Dror is with the Academic College of Tel-Aviv-Yaffo, Israel.

Joachim M. Buhmann is with ETH Zurich, Switzerland.

¹The capacity of a model may, in simple cases, be approximated by the number of free parameters.

II. CHALLENGE PROTOCOL

The challenge started September 30th, 2005 and ended March 1st, 2006 (duration: 21 weeks). Therefore, the competitors had several months to build classifiers with provided (labeled) training data. A web server was set up to submit prediction results on additional unlabeled data. Two unlabeled datasets were used for evaluation: a small validation set used during the development period and a very large test set for the final evaluation and the ranking of the participants. During a development period, the validation set performance was published immediately upon submission of prediction results. The test set performance remained undisclosed until the end of the competition. The labels of the validation set were published 30 days before the end of the competition. It was compulsory to return results on all five datasets to enter the final ranking. The number of submissions per participant was unlimited, but only the five last “complete” submissions were included in the final ranking.

In Figure 1, we show the performance evolution over time on the five datasets described in Table I. While the number of competitors kept increasing and the performance on the validation set kept improving, the performance of the test set improved mostly in the first 45 days of the challenge.³ This indicates that some participants may have overfitted the validation set.

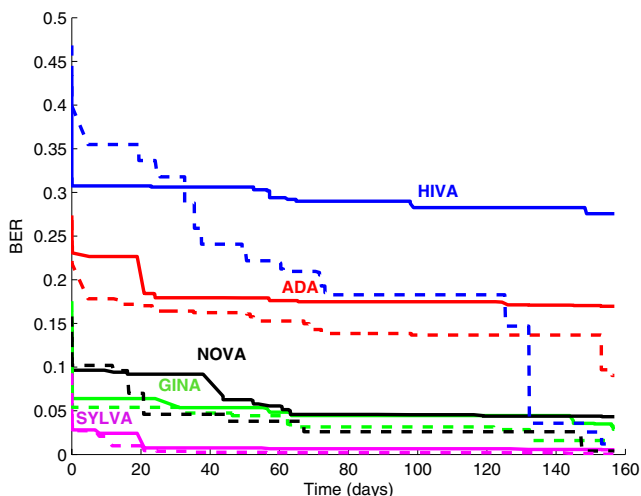


Fig. 1. Improvement of performance over time on the five tasks of the challenge. We show the best Balanced Error Rate (BER) achieved so far at any point in time. Solid line: Test set performance. Dashed line: Validation set performance.

In addition to providing predicted labels, the participants had to also provide an estimate of their performance on the test set. This guessed value allowed us to determine how well they controlled the overfitting problem, which is a critical aspect of good model selection. In Figure 2, we show the

³The drop in validation BER 30 days before the end of the challenge corresponds to the release of the validation set labels.

guessed performance as a function of the actual test set performance. Clearly, most people are overly optimistic.

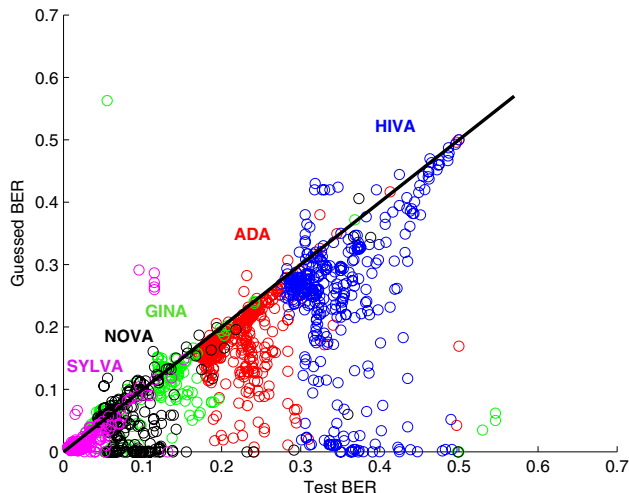


Fig. 2. Correlation between guessed performance and actual performance on the test set. Performance is measured in Balanced Error Rate, BER.

III. PERFORMANCE EVALUATION

Performance is measured in balanced error rate (BER), which is the average of the error rate on the positive class and the error rate on the negative class. As is known, for i.i.d. errors corresponding to Bernoulli trials with a probability of error p , the standard deviation of the error rate E computed on a test set of size m is $\sqrt{p(1-p)/m}$. This result can be adapted to the balanced error rate. Let us call m_+ the number of examples of the positive class, m_- the number of examples of the negative class, p_+ the probability of error on examples of the positive class, p_- the probability of error on examples of the negative class, and E_+ and E_- the corresponding empirical estimates. Both processes generating errors on the positive or negative class are Bernoulli processes. By definition, the balanced error rate is $BER = (1/2)(E_+ + E_-)$, and its variance is $var(BER) = (1/4)(var(E_+) + var(E_-))$. The standard deviation of the BER using m_+ and m_- examples is therefore

$$\sigma = \frac{1}{2} \sqrt{\frac{p_+(1-p_+)}{m_+} + \frac{p_-(1-p_-)}{m_-}}. \quad (1)$$

For sufficiently large test sets, we may substitute p_+ by E_+ and p_- by E_- to compute σ .

To rank the participants we adopted the following scheme: The entries are first ranked for each individual dataset. Then a global ranking is based on the average rank over all five datasets. For each individual dataset, the ranking score balances the classification accuracy and performance prediction accuracy. Denoting as BER the balanced error rate actually computed from predictions made on test examples,

and BER_{guess} the challenger's own performance prediction, we define our ranking score as:

$$S = BER + \delta_{BER}(1 - e^{-\delta_{BER}/\sigma}), \quad (2)$$

where $\delta_{BER} = |BER_{guess} - BER|$ measures in absolute value the difference between the computed BER and the predicted BER . The multiplicative factor $(1 - e^{-\delta_{BER}/\sigma})$ accounts for our uncertainty of the exact BER , since we can only estimate it on a finite test set of size m . If the BER error bar σ is small compared to the error of the challenger δ_{BER} , then this factor is just one. The ranking score becomes simply $BER + \delta_{BER}$. But if σ is large relative to the error made by the challenger, we have $S \simeq BER$. In Figure 3, we plot the ranking score S as a function of the predicted BER , BER_{guess} , for the numerical values $BER = 0.2$ and $\sigma = 0.05$.

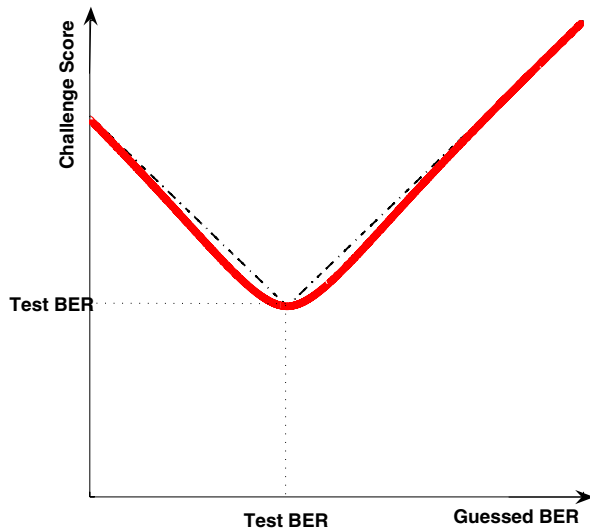


Fig. 3. Ranking score. The solid line is the ranking score as a function of the predicted BER , for an actual BER of 0.2 and an error bar $\sigma = 0.05$. The dashed line is $BER + \delta_{BER}$.

The area under the ROC curve (AUC) was also be computed. But the relative strength of classifiers was judged only on the BER .

IV. EXPERIMENTAL DESIGN

The data sets were chosen to span a variety of domains (drug discovery, ecology, handwritten digit recognition, text classification, and marketing), but their identity was not revealed until the end of the challenge so that no unfair advantage could be gained by those familiar with the application domain or the particular data. A report on the data preparation is now available [10]. We chose data sets that had sufficiently many examples to create a large enough test set to obtain statistically significant results. The input variables are continuous or binary, sparse or dense. All problems are

two-class classification problems. The data characteristics are summarized in Table I. “Sparsity” indicates the fraction of zeros in the data matrix; “type” indicates the input variable types (continuous, binary, or mixed; categorical variables were encoded with binary variables); “FracPos” indicates the fraction of examples of the positive class; “Tr/FN” indicates the ratio of number of training examples to number of features (input variables); “Fnum” indicates the number of features; and “TeNum” is the number of test examples.

Preparing the data included the following steps:

- Preprocessing to obtain features in the same numerical range (0 to 999 for continuous data and 0/1 for binary data).
- Randomizing the order of the patterns and the features to homogenize the data and further concealing its origin.
- Splitting the data into training, validation and test set of respective sizes m_{tr} , m_{va} , and m_{te} . The validation set is 100 times smaller than the test set. The training set is ten times larger than the validation set (and ten times smaller than the test set), which determines the sizes as $m_{va} = m/111$, $m_{tr} = m/11.1$, and $m_{te} = m/1.11$.

The motivation of the proportions of the training, validation and test data is that, according to Equation 1, if the number of examples on the validation set is 100 times smaller than that of the test set, the precision obtained for the BER using validation data will be ten times less precise. The training set being ten times larger than the validation set, this ratio gives an incentive to participants to find a more clever way of assessing the BER than just using the validation set results posted on the web site for their submission, during the development period.

TABLE I
DATASETS OF THE CHALLENGE IN PERFORMANCE PREDICTION.

Dataset	Sparsity	Type	FracPos	Tr/FN	FNum	TeNum
ADA	79.4%	mixed	24.8%	86.4	48	41471
GINA	69.2%	cont.	49.2%	3.25	970	31532
HIVA	90.9%	binary	3.5%	2.38	1617	38449
NOVA	99.7%	binary	28.5%	0.1	16969	17537
SYLVA	77.9%	mixed	6.2%	60.58	216	130858

In Figure 4, we show the distribution of performance on the test set of the entries who qualified for the final ranking. We see that the datasets vary in difficulty. HIVA (drug discovery) seems to be the most difficult dataset: the average BER and the spread are high. ADA (marketing) is the second hardest. The distribution is very skewed and has a heavy tail, indicating that a small group of methods “solved” the problem, which was not obvious to others. NOVA (text classification) and GINA (digit recognition) come next. Both datasets have classes containing multiple clusters. Hence, the problems are highly non-linear. This may property of the data explain the very long distribution tails. Finally, SYLVA (ecology) is the easiest dataset, due to the large amount of training data.

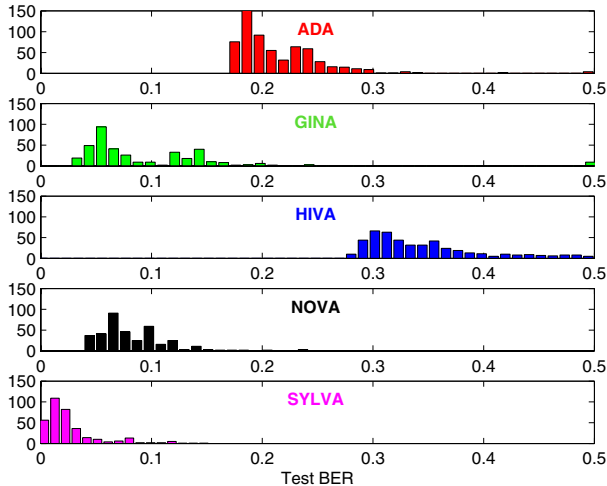


Fig. 4. Distribution of performance on the test set. Performance is measured in Balanced Error Rate, BER (see Section III).

V. MODELS PROVIDED

To trigger interest into the challenge, we provided initial baseline results with classical methods. Those methods were implemented with the Spider package developed at the Max Planck Institute for Biological Cybernetics [11]. Our package of methods implemented for the challenge called CLOP (Challenge Learning Object Package) is briefly described in the appendix. We made the CLOP software available to the participants, but the participants were not forced to use the provided package. The package may be downloaded from <http://clopinnet.com/isabelle/Projects/modelselect/Clop.zip> and documentation in the form of FAQ is found at <http://clopinnet.com/isabelle/Projects/modelselect/MFAQ.html>. Examples of application to the feature selection challenge are found at http://clopinnet.com/isabelle/Projects/ETH/Feature_Selection_w_CLOP.html.

The participants ended up using their own code rather than CLOP, except for one (Juha Reunanen), who ranked 16th in the challenge. If several competitors had elected to use CLOP we could have analyzed the impact of various model selection strategies in this controlled environment. However, the freedom given to the participants allowed them to reach better performance. We believe that this freedom also contributed to the high level of participation, which is key to deriving significant conclusions. We intend to organize a “post-challenge” game in which participants can compete on the same datasets (re-shuffled and re-partitioned), with the constraints of using CLOP models only.

VI. RESULTS

The results of the final ranking are reported in Table II, which includes the best entries of each ranked participant.

TABLE II

TOP RANKED CHALLENGE ENTRANTS (MEAN PERF. OVER THE 5 TASKS).

Entrant	Best entry	\overline{BER}	\overline{S}	\overline{Rank}
R. Lutz	LB tree m. c. a.	0.1090	0.1165	6.2
G. Cawley	Final #2	0.1124	0.1152	7.6
R. Neal	Bayesian N. N.	0.1118	0.1235	7.8
C. Dahinden	RF	0.1158	0.1264	7.8
Wei Chu	SVM/GPC	0.1153	0.1233	8.2
N. Meinshausen	ROMA	0.1167	0.1274	8.6
M. Boullé	SNB(CMA)+10kF	0.1307	0.1399	10.4
Torkkola & Tuv	ACE+RLSC	0.1191	0.1398	14
O. Chapelle	SVM-LOO	0.1262	0.1414	15.8
J. Wichard	submission 13	0.1252	0.1430	16.6
AAM, INTEL	IDEAL	0.1366	0.1644	16.6
V. Nikulin	GbO+MVf+SVM2	0.1334	0.1624	16.8
E. Harrington	ba4	0.1394	0.1492	16.8
Kai	Chi	0.1315	0.1559	17.6
Yu-Yen Ou	svm+ica	0.1273	0.1610	17.8
J. Reunanen	CLOP-models-5	0.1468	0.1545	19
Y.-J. Oyang	RBF+ICA(3:1)86+v	0.1324	0.1533	19
P. Haluptzok	NN Vanilla	0.1396	0.1634	19.2
T. Glasmachers	KTA+CV+SVM(3)	0.1488	0.1642	19.6
D. T.-H. Chang	PCA+ME+SVM+v.	0.1503	0.1689	21.4

TABLE III

TOP CHALLENGE RESULTS ACCORDING TO OUR RANKING SCORE S. THE CORRESPONDING ENTRIES ARE SHOWN IN TABLE IV.

Dataset	BER	AUC	σ	δ_{BER}	weight	S
ADA	0.1723	0.9149	0.0021	0.0073	0.9664	0.1793
GINA	0.0288	0.9712	0.0009	0.0017	0.8297	0.0302
HIVA	0.2757	0.7671	0.0068	0.0065	0.6145	0.2797
NOVA	0.0445	0.9914	0.0018	0.0009	0.3927	0.0448
SYLVA	0.0061	0.9991	0.0004	0.0001	0.2888	0.0062

The table shows the balanced error rate, averaged over all five datasets, the average score, and the average rank. The winner by average rank is Roman Lutz. The best average score was obtained by Gavin Cawley, who obtained also the best guessed BER. Radford Neal obtained the best AUC (data not shown). The full result tables are found on the web-site of the challenge (<http://www.modelselect.inf.ethz.ch/>).

We also show in Tables III and IV the best entries for each dataset. We note that several entrants who did well on individual datasets did not perform well overall, and vice versa. As indicated in the table, the top ranking entries have made errors on their performance prediction of the same order of magnitude as the error bar of the performance computed on test examples. This is an important achievement considering that the training set is ten times smaller than the test set and the validation set 100 times smaller.

TABLE IV

WINNERS BY DATASET.

Dataset	Entrant	Entry
ADA	Marc Boullé	SNB(CMA)+10kF(2D)tv
GINA	Kari Torkkola & Eugene Tuv	ACE+RLSC
HIVA	Gavin Cawley	Final #3 (corrected)
NOVA	Gavin Cawley	Final #1
SYLVA	Marc Boullé	SNB(CMA)+10kF(3D)tv

As per Equation 2, the ranking score is computed as: $S = BER + weight \delta_{BER}$, where $weight = (1 - e^{-\delta_{BER}/\sigma})$. Since the error δ_{BER} made by the best challengers in predicting their performance is of the same order of magnitude as the error bar of the BER , σ , their penalty for wrong estimations (modulated by the weight shown in the table) is moderate and S is relatively close to BER . However, we verified that only a few percent of the entries had a δ_{BER} better than σ . Therefore the weight of δ_{BER} is generally close to one, and the score close to $S = BER + \delta_{BER}$. The influence of the two terms of the score S can be modulated by $weight = (1 - e^{-\gamma \delta_{BER}/\sigma})$ using a varying parameter γ . We observed by plotting the score as a function of γ that the curves cross for small values of γ but never beyond $\gamma = 1$. Smaller values of γ would diminish the influence of δ_{BER} , which is not desirable for the purpose of benchmarking performance prediction. Values larger than one would not change the results. This justifies in retrospect our choice $\gamma = 1$. In Figure 5, we show an example of such curves for the entries of the top ranking participants for the dataset GINA. Clearly, some participants did better than others at guessing their test BER as reflected by the crossing of the curves as the influence of δ_{BER} increases. The challenge score rewards better BER predictions.

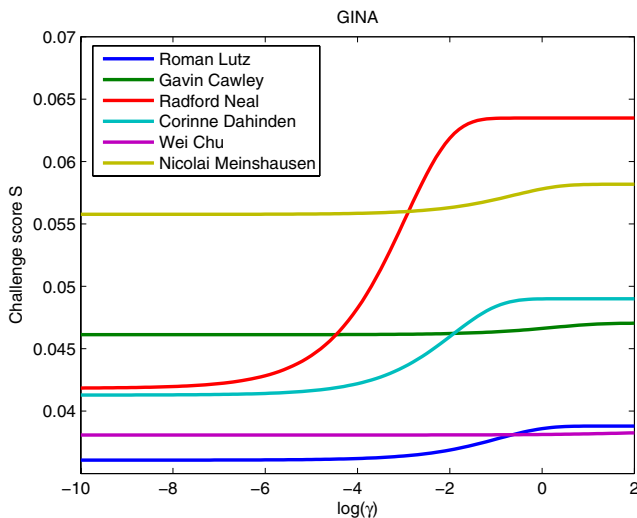


Fig. 5. Monitoring the balance between BER and δ_{BER} . We illustrate on the GINA dataset the influence of γ in the formula $S = BER + (1 - e^{-\gamma \delta_{BER}/\sigma}) \delta_{BER}$. The challenge score corresponds to $\gamma = 1$.

VII. SURVEY OF THE METHODS USED

We surveyed the participants to determine what method they used for preprocessing, classification, model selection, and prediction of the test BER . We summarize in this section the results of this survey. More details can be found in the participants papers published in the WCCI 2006 proceedings.

We particularly examined how the various methods did with respect to optimizing the test BER and the δ_{BER} . In

Figure 6 each point represents one of the 117 final entries, for each dataset. The best ranking entry according to the challenge score is indicated by an arrow. The symbols code for the methods used:

- X: Mixed or unknown method.
- TREE: Ensembles of trees (like Random Forests, RF).
- NN/BNN: Neural networks or Bayesian neural nets.
- NB: Naïve Bayes.
- LD/SVM/KLS/GP: Methods linear in their parameters, including kernel methods and linear discriminant (LD), Support Vector Machines (SVM), Kernel Least-Squares (KLS) and LS-SVM, Gaussian Processes (GP).

Preprocessing and feature selection

The survey reveals that the preprocessing methods used are very elementary:

- Tree classifiers most often use no preprocessing at all (Roman Lutz, Corinne Dahinden, Intel).
- The most common preprocessing is feature standardization (subtract the mean and divide by the standard deviation for each feature).
- The naïve Bayes method (Marc Boullé) uses discretization of continuous variables and a construction of extra features as a sum of elementary features to alleviate independence assumptions.
- A few entries used PCA or ICA to extract features.
- Most entries used no feature selection at all: they relied on regularization for overfitting avoidance. Some entries used embedded feature selection (ARD priors for the Bayesian neural network – Radford Neal, feature scaling for SVMs – Olivier Chapelle); a few used univariate filters; the naïve Bayes (Marc Boullé) used a wrapper feature selection; the entry of Torkkola and Tuv used the features generated by an ensemble of tree classifier and computed a threshold of significance using “random probes”.
- Some entries resampled the training data to balance the two classes.

There does not seem to be a correlation between the type of preprocessing used and how well people did in the challenge.

Classifiers

In this challenge, many classification methods did well. There is a variety of methods in the top ranking entries:

- Ensembles of decision trees (Roman Lutz, Corinne Dahinden, Intel).
- Kernel methods/SVMs (Gavin Cawley, Wei Chu, Kari Torkkola and Eugene Tuv, Olivier Chapelle, Vladimir Nikulin, Yu-Yen Ou).
- Bayesian Neural Networks (Radford Neal).
- Ensembles of linear methods (Nicolai Meinshausen).
- Naïve Bayes (Marc Boullé).

Others used mixed models. It is interesting to note that single methods did better than mixed models.

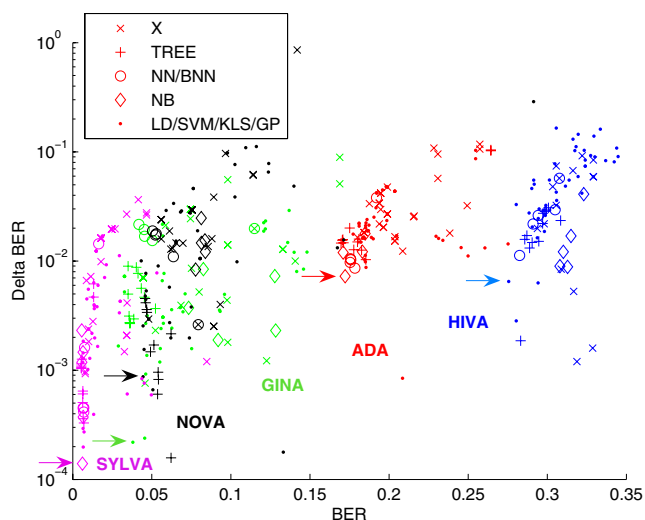


Fig. 6. Results on the test set of all final entries. The best entries for each dataset are indicated with an arrow.

Figure 6 reveals that Naïve Bayes did very well on two datasets (ADA and SYLVA) but poorly on others. Similarly, kernel methods did very well on most datasets (they rank first for HIVA, GINA, and NOVA), but they did poorly on ADA. This failure on ADA make them rank only fifth in the overall ranking. They are the most frequently used type of method, but their performance shows a lot of variance. On the contrary, ensembles of decision trees are not so popular, but they perform consistently well on all datasets in this challenge, even though they are never the top entry for any of the datasets. The challenge winner used an ensemble of decision trees. Similarly, Bayesian neural networks did well on all datasets, even though they were not best for any. They end up ranking third in the overall scoring.

This analysis also tells us something about the datasets: SYLVA has a large number of training examples, so all methods essentially perform well, even the simple naïve Bayes. We know by design that GINA and NOVA are very non-linear. The top ranking participants used highly non-linear methods and naïve Bayes failed. HIVA seems to also be in this category, not too surprisingly: chemical molecules with *very* different structures can be active or inactive drugs. ADA has a particularity that makes kernel methods fail and which should be further investigated. We conjecture that is could be because the variables are mixed categorical/binary/continuous.

Model selection and performance prediction

Even though it is interesting to see how well methods performed as a function of the classification techniques, the most interesting thing is to analyze the methods of prediction of the BER and the methods of model selection, since this was the theme of the challenge. We can roughly categorize the methods used as follows:

- Nested cross-validation loops.** The entrant who obtained the best average guess error (Gavin Cawley, average guess error: 0.0034) used a rather sophisticated cross-validation scheme. He adjusted hyperparameters with a “virtual leave-one-out” cross-validation (VLOO). For regularized least-square kernel classifiers (LS-SVMs, kernel ridge regression, RLSC, Gaussian processes), it is possible to compute the leave-one-out error without training several classifiers (each time leaving one example out). Only one training with the entire training set and some inexpensive additional calculations are required. Gavin Cawley explored various loss-functions for the VLOO method using LS-SVMs for classification. He selected the best loss function with an outer loop of cross-validation (drawing 100 random 90%training-10%validation splits; we call this 100CV). After selecting his model, he re-estimated performance by 100CV using fresh data splits. The entrant who obtained the second best average guess error (Juha Reunanen, average guess error: 0.0048) performed a similar type of cross-validation called “cross-indexing”, which uses nested cross-validation loops. The BER guess was obtained by the cross-validation performance in the outer loop. Nested cross-validation loops can be expensive computationally, however, in the case of the use of VLOO, the computational increase is minimal. We note however that VLOO is not available for all methods. Approximate formulas are available for SVMs (as used by Olivier Chapelle, average guess error: 0.0137) and neural networks.
- Plain cross-validation.** Many participants used plain cross-validation, with a preference for 10-fold cross-validation. They chose their hyperparameters on the basis of the smallest cross-validated BER. The same cross-validated BER (CV BER) was used to guess the test BER, hoping that the bias introduced by using only 90% of the training data would be compensated by the bias introduced by selecting the model having smallest CV BER. This strategy seems to have been reasonable since the challenge winner (Roman Lutz, average guess error: 0.0059) used it. He won because of his good BER performance. The second best entrant (Gavin Cawley) had better BER guesses (average guess error: 0.0034). A few entrants took care of balancing the fraction of positive and negative examples in the data splits to reflect the proportions found in the entire training set (stratified cross-validation).
- Other methods.** A few entrants performed model selection or estimated the performance on future data using training data, without reserving validation data or performing cross-validation, which is possible for regularized and Bayesian methods not prone to overfitting. This was used as a model selection strategy for the naïve Bayes classifier (Marc Boullé). Radford Neal for his Bayesian Neural Network predicts the error

rate using training data only, but this was not one of the best performing methods (average guess error: 0.0122). Other methods include the use of performance bounds for SVM model selection like the Radius Margin bound (Olivier Chapelle). Bagging methods (including Random Forests) use bootstrap resampling. The final model makes decisions according to a vote of the models trained on the various bootstrap samples. The error rate of the model can be estimated using the “out-of-bag” samples. This method was used by Nicolai Meinshausen (average guess error: 0.0098). The least reliable method was to use the validation set of the challenge to predict the test set performance. The best ranking participants did not use this method.

VIII. CONCLUSION

This paper presented the design and results of the WCCI 2006 competition in performance prediction. The challenge was very successful in attracting a large number of participants who delivered many interesting ways of approaching performance predictions. We observed that rather unsophisticated methods (e.g. simple 10-fold cross validation) did well to predict performance. Nested cross-validation loops are advisable to gain extra prediction accuracy. They come at little extra computational expense, if the inner loop uses virtual leave-one-out. Successful model selection techniques include cross-validation and regularization methods. Ensemble and Bayesian methods provide an efficient alternative to model selection by constructing committees of classifiers. A number of sophisticated methods with appealing theoretical motivations were proposed for the model selection special session, but the authors did not compete in the challenge. We believe there is still a gap to be filled between theory and practice in this game of performance prediction and model selection.

ACKNOWLEDGMENTS

The organization of this challenge was a team effort to which many have participated. We are particularly grateful to Olivier Guyon (MisterP.net) who implemented the back-end of the web site. The front-end follows the design of Steve Gunn (University of Southampton), formerly used for the NIPS 2003 feature selection challenge. We are thankful to Bernd Fischer (ETH Zurich) for administering the computer resources. Other advisors and beta-testers are gratefully acknowledged: Yoshua Bengio (University of Montreal), Asa Ben-Hur (Colorado State university), Lambert Schomaker (University of Groningen), and Vladimir Vapnik (NEC, Princeton). The Challenge Learning Object Package (CLOP) is based on code to which many people have contributed: The creators of the spider: Jason Weston, Andre Elisseeff, Gikhhan Bakir, Fabian Sinz. The developers of the packages attached: Chih-Chung Chang and Chih-JenLin Jun-Cheng (LIBSVM), Chen, Kuan-Jen Peng, Chih-Yuan Yan, Chih-Huai Cheng, and Rong-En Fan (LIBSVM Matlab interface),

Junshui Ma and Yi Zhao (second LIBSVM Matlab interface), Leo Breiman and Adele Cutler (Random Forests), Ting Wang (RF Matlab interface), Ian Nabney and Christopher Bishop (NETLAB), Thorsten Joachims (SVMLight), Ronan Collobert (SVM Torch II), Jez Hill, Jan Eichhorn, Rodrigo Fernandez, Holger Froehlich, Gorden Jemwa, Kiyoungh Yang, Chirag Patel, Sergio Rojas. This project is supported by the National Science Foundation under Grant N0. ECS-0424142. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Predicant biosciences, Microsoft, and Unipen provided additional support permitting to grant prizes to the winners.

REFERENCES

- [1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer series in statistics. New York: Springer, 2001.
- [2] A. Webb, *Statistical Pattern Recognition*, 2nd ed. Chichester, West Sussex, UK: John Wiley & Sons, Inc., 2002.
- [3] V. Vapnik, *Statistical Learning Theory*. N.Y.: John Wiley and Sons, 1998.
- [4] M. J. Kearns, Y. Mansour, A. Y. Ng, and D. Ron, “An experimental and theoretical comparison of model selection methods,” in *Computational Learning Theory*, 1995, pp. 21–30. [Online]. Available: citeseer.ist.psu.edu/article/kearns95experimental.html.
- [5] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *IJCAI*, 1995, pp. 1137–1145. [Online]. Available: citeseer.ist.psu.edu/kohavi95study.html.
- [6] A. M. Molinaro, R. Simon, and R. M. Pfeiffer, “Prediction error estimation: a comparison of resampling methods,” *Bioinformatics*, vol. 21, no. 15, pp. 3301–3307, 2005.
- [7] C. B. D.J. Newman, S. Hettich and C. Merz, “UCI repository of machine learning databases,” 1998. [Online]. Available: <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
- [8] D. Kazakov, L. Popelinsky, and O. Stepankova, “MLnet machine learning network on-line information service. [Online]. Available: <http://www.mlnet.org>.
- [9] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, “Result analysis of the nips 2003 feature selection challenge,” in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. Cambridge, MA: MIT Press, 2005, pp. 545–552.
- [10] I. Guyon, “Experimental design of the WCCI 2006 performance prediction challenge,” Clopinet, Tech. Rep., 2005. [Online]. Available: <http://clopinet.com/isabelle/Projects/modelselect/Dataset.pdf>.
- [11] J. Weston, A. Elisseeff, G. Bakir, and F. Sinz, “The Spider machine learning toolbox, 2005. [Online]. Available: <http://www.kyb.tuebingen.mpg.de/bs/people/spider/>.
- [12] C. M. Bishop, *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press, 1996.
- [13] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
- [14] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, [Online]. Available: <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [15] T. R. G. et al., “Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring,” *Science*, vol. 286, pp. 531–537, 1999.
- [16] K. Kira and L. A. Rendell, “A Practical Approach to Feature Selection,” in *Proceedings of the 9th International Conference on Machine Learning, ICML'92*. San Francisco, CA: Morgan Kaufman, 1992, pp. 249–256.
- [17] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar, “Ranking a random feature for variable and feature selection,” *Journal of Machine Learning Research*, vol. 3, pp. 1399–1414, 2003.

CLOP IN A NUTSHELL

A. Learning objects

The Spider package on top of which our package is built, uses Matlab objects (The MathWorks, <http://www.mathworks.com/>). Two simple abstractions are used:

- **data:** Data objects include two members X and Y, X being the input matrix (patterns in lines and features in columns), Y being the target matrix (i.e. one column of +1 for binary classification problems).
- **algorithms:** Algorithm objects representing learning machines (e.g. neural networks, kernel methods, decision trees) or preprocessors (for feature construction, data normalization or feature selection). They are constructed from a set of hyper-parameters and have at least two methods: train and test. The train method adjusts the parameters of the model. The test method processes data using a trained model.

For example, you can construct a data object D:

```
> D = data(X, Y);
```

The resulting object has 2 members: D.X and D.Y. Models are derived from the class algorithm. They are constructed using a set of hyperparameters provided as a cell array of strings, for instance:

```
> hyperparam = {'h1=val1', 'h2=val2'};
> model0 = algorithm(hyperparam);
```

In this way, hyperparameters can be provided in any order or omitted. Omitted hyperparameters take default values.

To find out about the default values and allowed hyperparameter range, one can use the “default” method:

```
> default(algorithm)
```

The constructed model model0 can then be trained and tested:

```
> [Dout, model1] = train(model0, Dtrain);
> Dout = test(model1, Dtest);
```

model1 is a model object identical to model0, except that its parameters (some data members) have been updated by training. Matlab uses the convention that the object of a method is passed as first argument as a means to identify which overloaded method to call. Hence, the “correct” train method for the class of model0 will be called. Since Matlab passes all arguments by value, model0 remains unchanged. By calling the trained and untrained model with the same name, the new model can overwrite the old one. Repeatedly calling the method “train” on the same model may have different effects depending on the model.

To save the model is very simple since Matlab objects know how to save themselves:

```
> save('filename', 'modelname');
```

This feature is very convenient to make results reproducible, particularly in the context of a challenge.

B. Compound models: chains and ensembles

The Spider (with some CLOP extensions) provides ways of building more complex “compound” models from the basic algorithms with two abstractions:

- **chain:** A chain is a learning object (having a train and test method) constructed from an array of learning objects. Each array member takes the output of the previous member and feeds its outputs to the next member.
- **ensemble:** An ensemble is also a learning object constructed from an array of learning objects. The trained learning machine performs a weighted sum of the predictions of the array members. The individual learning machines are all trained from the same input data. The voting weights are set to one by default. An interface is provided for user-defined methods of learning the voting weights.

Compound models behaves like any other learning object: they can be trained and tested. In the following example, a chain object cm consists of a feature standardization for preprocessing followed by a neural network:

```
> cm=chain({standardize, neural});
```

While a chain is a “deep” structure of models, an ensemble is a “wide” structure. The following command creates an ensemble model em:

```
> em=ensemble({neural, kridge, naive});
```

To create more complex compound models, models of the same class with different hyperparameters or different models can be combined in this way; chains can be part of ensembles or ensembles can be part of chains.

C. Model selection

The Spider provides several objects for cross-validation and other model selection methods. The challenge participants may elect to use those built-in objects or write their own. A model selection object is derived from the class “algorithm” and possesses train and test methods. For instance, 10-fold cross-validation is performed as follows:

```
> cv_model=cv(my_model, {'folds=10'});
> cv_output = train(cv_model, Dtrain);
```

D. Getting started

On the web-site of the challenge (<http://www.modelselect.inf.ethz.ch/models.php>), sample code is provided including examples of combinations of modules into chains and ensembles, and a script to load data, train a model, and save the model and the results into an archive ready to upload.