Performance Prediction Challenge

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

Abstract— **We have organized for WCCI 2006 a challenge in performance prediction. 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). By "performance prediction" we mean predicting how well a given classifier will perform on new unseen data. Over 100 participants have been trying to build from training data the best possible classifier and guess their generalization error on a large unlabeled test set. The outcome of this challenge will help answering such questions as: Is cross-validation the best solution? What should k be in k-fold? Can one use theoretical performance bounds to better assess generalization? The challenge is available at http://www.modelselect.inf.ethz.ch/. It will end March 1^{st} 2006 and the results will be discussed at the special section on model selection at the WCCI 2006 conference. This paper summarizes the design of the benchmark; the final version of the paper will include the results of the challenge (not yet available since the challenge is not over).**

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 raises the problem of making best use of a limited amount of available data to both train the models and assess their performances.

The problem of predicting generalization performance has 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 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 approach has been the development of 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], make it possible to conduct systematic studies. Yet, no consensus seems to be emerging, possibly due to the bias each team may be introducing 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). This motivates our present effort.

Participants to the challenge will submit papers to the special session on model selection where the results of the challenge will be discussed. The topic of the special session has been chosen to be slightly broader of that of the challenge, but closely related. Indeed, model selection is rendered trivial if the performances of the models under scrutiny are known with precision.

Our challenge design is inspired by those of previous competitions, and particularly by the challenge in feature selection that we co-organize 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] Entrants must provide 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 may submit results on a validation subset of the data to obtain immediate feed-back. The final ranking will be performed on a separate test set.

II. Challenge protocol

The challenge started September 30^{th}, 2005 and will end March 1^{st}, 2006. Therefore, the competitors have several

Isabelle Guyon is an independent consultant, Berkeley, California, USA, presently a visiting professor at ETH Zurich (isabelle@clopinet.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 Buhmann is with ETH Zurich, Switzerland.

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

[2]It 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.

months to build classifiers with provided (labeled) training data. A web server is provided to submit prediction results on additional unlabeled data. Two unlabeled datasets are used for evaluation: a small validation set used during the development period and a very large test set to do the final evaluation and the ranking of the participants. During a development period, the validation set performance is published immediately upon submission of prediction results. The test set performance remains undisclosed until the end of the competition. The labels of the validation set are published shortly before the end of the competition.

In addition to providing predicted labels, the participants must also provide an estimate of their performance on the test set. The participants must return results on all five datasets to enter the final ranking. The number of submissions per participant is unlimited, but only the five last "complete" submissions will be included in the final ranking.

III. Performance evaluation

Performances are 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 Bernouilli 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 Bernouilli 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

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

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 competing predictions are first ranked for each individual dataset. Then global ranking is obtained by using the sum of the ranks on the five dataset as a ranking score. For each individual dataset, the ranking score used balances the accuracy of classification and the accuracy of performance prediction. Denoting as BER the balanced error rate actually computed from predictions made on test examples, and BER_{pred} the prediction made by the challenger of his performance, we define our ranking score as:

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

where $\delta_{BER} = |BER_{pred} - BER|$ 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 uncertainly 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 relatively to the error made by the challenger, we have $R \simeq BER$. In Figure 1, we plot the ranking score R as a function of the predicted BER, BER_{pred}, for the numerical values $BER = 0.2$ and $\sigma = 0.05$.

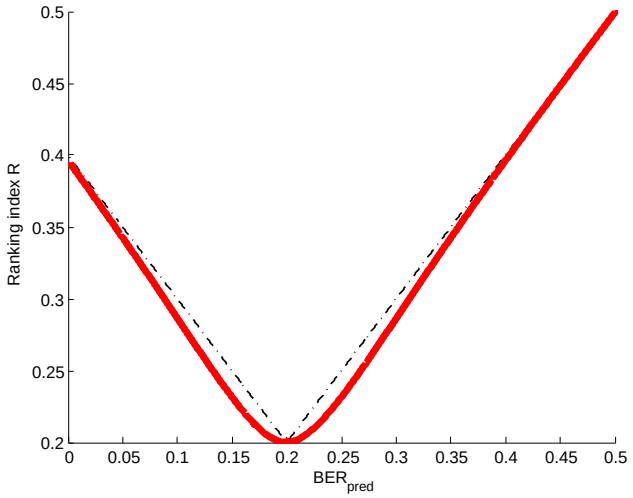

Fig. 1. Ranking score. The red curve is the ranking score as a function of the predicted BER, for an actual BER of 0.2 and an error bar $\sigma = 0.05$. The dashed line is $BER + \delta_{BER}$.

The area under the ROC curve (AUC) will also be computed, if the participants provide classification confidences (the .conf files) in addition to class label predictions (the compulsory .resu files). But the relative strength of classifiers will be judged only on the BER. Other statistics may also be computed and will be reported (e.g. performances using other loss functions) but will not be used towards determining the winner.

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 will not be revealed until the end of the challenge so that no unfair advantage can be gained by those familiar with the application domain or the particular data. 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 time 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 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 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 WCCI06 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

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 [10]. Our package of methods implemented for the challenge is called CLOP (Challenge Learning Object Package). 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://clopinet.com/isabelle/Projects/modelselect/Clop.zip` and documentation in the form of FAQ is found at `http://clopinet.com/isabelle/Projects/modelselect/MFAQ.html`.

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 train 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 a also 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 create 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. Objects provided for the challenge

For the purpose of the challenge, we provided a number of modules, chosen because those methods performed well in the previous challenge we organized [9]. There are five classifiers:

- **kridge** Kernel ridge regression.
- **naive** Naive Bayes classifier.
- **neural** Neural network (using Netlab [11]).
- **rf** Random Forest (using the original implementation [12]).
- **svc** Support vector classifier (using LibSVM [13]).

We purposely limited the number of hyperparameters of each method to avoid making the model selection problem so complex that solving it would require resorting to intuitions or heuristics to prune first the space of hyperparameters. For instance, the two kernel methods "kridge" and "svc" use a single kernel: $k(\mathbf{x}, \mathbf{x}') = (\mathbf{coef_0} + \mathbf{x} \cdot \mathbf{x}')^{\mathbf{d}} \mathbf{exp}(-\gamma ||\mathbf{x} - \mathbf{x}'||^{\mathbf{2}})$. In addition to the three hyperparameters $coef_0$, d, and γ, the kernel methods have a "shrinkage" parameter for regularization, corresponding to a small value added to the diagonal of the kernel matrix.

The challenge modules also include five preprocessors:

- **`standardize`**: Standardization of the features (the columns of the data matrix are divided by their standard deviation; optionally, the mean is first subtracted if center=1.)
- **`normalize`**: Normalization of the lines of the data matrix (optionally the mean of the lines is subtracted first.)
- **`shift_n_scale`**: Performs $\mathbf{x} \leftarrow (\mathbf{x} - \mathbf{offset})/\mathbf{scale}$ globally on the data matrix. Optionally performs in addition $log(1 + \mathbf{x})$.
- **`pc_extract`**: Extraction of features with principal component analysis.
- **`subsample`**: Takes a subsample of the training patterns. May be used to downsize the training set or exclude outliers.

Finally, we also provide five feature selection methods:

- **s2n**: Signal-to-noise ratio coefficient for feature ranking [14].
- **relief**: Relief ranking criterion [15].
- **gs**: Forward feature selection with Gram-Schmidt orthogonalization [16].
- **rffs**: Random Forest used as feature selection filter [].
- **svcrfe**: Recursive Feature Elimination filter using svc [17].

All the feature selection methods chosen are either individual feature ranking methods or nested subset methods (forward selection or backward elimination). Hence, they all provide a ranking of features as part of the trained "model". The number of features to be selected is a hyperparameter, which may be varied after training to avoid recomputing the ranking.

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.

D. 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);
```

VI. Results

Although we cannot yet reveal who the winner is, in Table II we report anonymously the results on the various datasets, as of December 14, 2005. We call " weight" the factor $(1 - e^{-\delta_{BER}/\sigma})$ in Equation 2. As per Equation 2, the ranking score is computed as: $R = BER + weight\ \delta_{BER}$,

TABLE II

TOP MID-CHALLENGE RESULTS ACCORDING TO OUR RANKING SCORE R.

Dataset	BER	AUC	σ	δ_{BER}	weight	R
ADA	0.1747	0.9102	0.0026	0.0096	0.9751	0.1841
GINA	0.0445	0.9566	0.0012	0.0011	0.6001	0.0453
HIVA	0.3062	0.7089	0.0089	0.0023	0.2277	0.3067
NOVA	0.0458	0.9900	0.0021	0.0032	0.7821	0.0483
SYLVA	0.0065	0.9991	0.0006	0.0006	0.6321	0.0069

where $weight = (1 - e^{-\delta_{BER}/\sigma})$. We observe from the results that 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 therefore moderate and R is close to BER. Presently, all the top ranking participants according to R are also top ranked according to BER. However, in lower ranking participants, the two rankings differ.

VII. CONCLUSION

This paper presented the design of the on-going WCCI 2006 competition in performance prediction. Our final paper will include a description of the datasets (whose identity is presently unavailable to participants). The results of the competition will be analyzed and we will provide information about the best ranking entries. We will learn from the designs of the top ranking participants what should be done to efficiently perform model selection and accurately predict model performance. Presently, all the top ranking participants 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.

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 , Gikhan 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, Kiyoung 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 Learing 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/$\sim$mlearn/MLRepository.html

[8] D. Kazakov, L. Popelinsky, and O. Stepankova, "MLnet machine learning network on-line information service." in *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] J. Weston, A. Elisseeff, G. BakIr, and F. Sinz, "The Spider machine learning toolbox," *http://www.kyb.tuebingen.mpg.de/bs/people/spider/*, 2005.

[11] C. M. Bishop, *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press, 1996.

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

[13] 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.

[14] 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.

[15] 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 Kauffman, 1992, pp. 249–256.

[16] 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.

[17] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.