Combining Lazy Learning, Racing and Subsampling for Effective Feature Selection

Gianluca Bontempi, Mauro Birattari and Patrick E. Meyer
ULB, Université Libre de Bruxelles, Bruxelles - Belgium

Abstract

This paper presents a wrapper method for feature selection that combines Lazy Learning, racing and subsampling techniques. Lazy Learning (LL) is a local learning technique that, once a query is received, extracts a prediction by locally interpolating the neighboring examples of the query which are considered relevant according to a distance measure. Local learning techniques are often criticized for their limitations in dealing with problems with high number of features and large samples. Similarly wrapper methods are considered prohibitive for large number of features, due to the high cost of the evaluation step. The paper aims to show that a wrapper feature selection method based on LL can take advantage of two effective strategies: racing and subsampling. While the idea of racing was already proposed by Maron and Moore, this paper goes a step further by (i) proposing a multiple testing technique for less conservative racing (ii) combining racing with sub-sampling techniques.

1 Introduction

Lazy Learning (LL) is a local modeling technique which is *query-based* in the sense that the whole learning procedure (i.e. structural and parametric identification) is deferred until a prediction is required. In previous works we presented an original *Lazy Learning* algorithm [3, 1] that selects automatically on a query-by-query basis the optimal number of neighbors, and its application to data analysis, time series prediction, system identification and nonlinear control. Despite the large amount of applications to real tasks, local methods are often a target of criticism as far as computational requirements for dealing with large datasets (i.e. many variables and/or large samples) are taken into consideration.

This paper discusses how to effectively employ the LL algorithm for feature selection (for an up-to-date state of the art on feature selection see [7]). In particular, we will focus here on a wrapper method based on LL. In the wrapper approach [9] the feature subset selection algorithm exists as a wrapper around the learning algorithm, which is often considered as a black box able to return (e.g. via cross-validation) an evaluation of the quality of a feature subset. While most of the wrapper algorithms are defined independently of the learning machine, we propose here a method which takes advantage of the unique aspects of a local learning algorithm.

The usefulness of a local modeling approach for reducing the cost of feature selection was first presented in [10]. The idea consists in assessing a large number of feature subsets by performing cross-validation only on a reduced test set. On the basis of well-known statistical results, it is possible to show that families of good feature subsets can be rapidly found by quickly discarding the bad subsets and concentrating the computational effort on the better ones. This model selection technique was called the *Hoeffding race* by Maron and Moore [10], with reference to Hoeffding's formula which puts a bound on the accuracy of a sampled mean of N observations as an estimator of the expected value. Local modeling techniques fit well in this paradigm thanks to the property that modeling is performed only when a prediction is required. Then, by reducing the size of the test set, the computational cost required for assessment is reduced, too.

The aim of this work is: i) to speed up further the evaluation step of a wrapper method by reducing, together with the number of test predictions, also the size of the training set required for assessing the quality of a candidate feature set, ii) to improve the efficiency of the wrapper search by discarding more rapidly those candidates which, on the basis of the tests made till that moment, appear as significantly worse than others. In particular, the main two contributions of the paper are: the combination of racing and sub-sampling techniques with the Lazy Learning algorithm previously proposed by the authors and the use of a multiple testing criterion (the Friedman test) for performing a more statistically founded selection.

The use of sampling [8] enhances the performance of the racing approach to feature selection by making possible the definition of a number of lightweight experimental benchmarks to assess, compare and discriminate between candidate subsets. Each experimental bench-

mark relies on small training and test sets, which are obtained by randomly sub-sampling the available dataset. The idea is that, in order to discriminate between a very large number of candidates, the combination of multiple small and fast benchmarks can be more effective than a single big and time consuming validation procedure.

The second contribution of the paper lies in the use of a nonparametric multiple test, the Friedman test [4], to compare different configurations of input variables and to select the ones to be eliminated from the race. The use of the Friedman test for racing was proposed first by one of the authors in the context of a technique for comparing metaheuristics for combinatorial optimization problems [2]. This is the first time that the technique is used in a feature selection setting. Note that in some sense this method fills the gap between Hoeffding race [10] and BRACE [11]: similarly to Hoeffding race it performs a nonparametric test, and similarly to BRACE it considers a blocking design.

2 The RACSAM (racing+sub-sampling) algorithm

The RACSAM wrapper algorithm is based on the idea that once the evaluation is based on the LL algorithm,[1] this can be made faster by considering training and test sets of limited size.

The main steps of the RACSAM algorithm are: (i) the creation of a number of lightweight experimental benchmarks, each characterized by a small training and test set, (ii) the use of the experimental benchmarks to assess in parallel a large family of candidates, (iii) the discarding of those candidates resulting significantly worse than others.

2.1 The F-race algorithm

The racing algorithm proposed in this paper, F-Race in the following, takes as input a training set of size N, a test set of size N_{ts}, a set of M candidate feature sets $\mathcal{S}_m \subseteq \{1, \ldots, n\}$ (also called configurations) and a number $W \geq 1$ of expected winners. The expected output is a set of at most W configurations which are significantly better than the others.

Let $\Theta_0 = \{\mathcal{S}_m, m = 1, \ldots, M\}$ the set of configurations at the start of the algorithm. A racing algorithm proceeds by generating a sequence $\Theta_0 \supseteq \Theta_1 \supseteq \Theta_2 \supseteq \ldots$, of nested sets of candidate configurations. The step from a set Θ_{q-1} to Θ_q is obtained by possibly discarding some configurations that appear to be suboptimal on the basis of information available at step q. Let M_{q-1} the cardinality of Θ_{q-1}, that is the number of remaining configurations and $I_{q-1} = \{m : \mathcal{S}_m \in \Theta_{q-1}\}$ the indices of the configuration still competing before the execution of the qth step.

At the qth step, $1 \leq q \leq N_{ts}$, the racing algorithm computes for all $\mathcal{S}_m \in \Theta_{q-1}$

$$\hat{y}_{q,m} = LL(x_q, D_N, \mathcal{S}_m) \tag{1}$$

where $LL(\cdot)$ is the prediction returned by a LL algorithm where the feature set is $\mathcal{S}_m \in \Theta_{q-1}$ and the training set is made of N samples.

This value is used to fill an evaluation matrix E, sized $N_{ts} \times M$, whose generic $[q, m]$ term is

$$E[q, m] = |y_q - \hat{y}_{q,m}|, \quad q = 1, \ldots, N_{ts}, \quad m \in I_{q-1} \tag{2}$$

where y_q is the observed output for the input x_q. Step q terminates defining set Θ_q by dropping from Θ_{q-1} the configurations that appear to be suboptimal in the light of the statistical test described in the following section. Note that this test compares only the $M_{q-1} \leq M$ columns of the evaluation matrix, whose indices are in I_{q-1}, which represent the feature configurations still in the race.

The above described procedure is iterated and stops either when all configurations but W are discarded (i.e. $M_q \leq W$ for some q) or when $q = N_{ts}$. The advantage of racing is that in the first case, $M - W$ configurations are discarded by having recourse only to a subset of the N_{ts} samples available for testing.

The statistical test: the racing algorithm we propose in F-Race[2] is based on the Friedman test, a statistical method for hypothesis testing also known as Friedman two-way analysis of variance by ranks [4]. The null hypothesis of the test assumes that all remaining configurations in the race belong to the same error distribution. If at the qth step the null of the aggregate comparison is not rejected, all candidates in Θ_{q-1} pass to Θ_q. On the other hand, if the null is rejected, the configuration with the largest estimated mean-square error is discarded and the test repeated with $M_{q-1} - 1$ configurations, until the null hypothesis is not rejected.

The main merit of our nonparametric approach is that it does not require to formulate hypotheses on the distribution of the observations. A second role played by the Friedman test is to implement in a natural way a blocking design [5]. Blocking is an effective way for normalizing the costs observed on different conditions. By focusing only on the ranking of the different configurations within each condition, blocking eliminates the risks that the variation due to the difference among test samples washes out the variation due to the difference among configurations.

[1] We refer the reader to [3, 1] and to the publicly available R package `lazy` (http://cran.r-project.org/src/contrib/Descriptions/lazy.html) for a description of the main features of the LL algorithm.

[2] available at http://cran.r-project.org/src/contrib/Descriptions/race.html

2.2 The racing+sub-sampling combination

In spite of the improvement due to racing, evaluation based on LL may still appear computationally expensive if the training set is very large. It is sufficient to note that the complexity of the LL prediction in (1) is proportional to the number of samples in D_N.

The added value of sampling consists in using in (1) training sets $D_{\hat{N}}$, composed of $\hat{N} < N$ samples, which are a randomized subset of the original set D_N. The reduction of the training set size speeds up the LL prediction and consequently each step of the F-race algorithm.

Let us notice however that the computational gain occurs at the cost of the deterioration of the assessment of the quality of each single configuration. In terms of bias/variance trade-off sub-sampling implies variance increase. In general terms, the rationale for the RACSAM approach is that very bad models should be detected rapidly and with small effort by exploiting only a part of the information of the training set, reserving the intensive use of the entire information only to the most difficult cases. This is implemented in practice by increasing gradually the size $\hat{N}$ of the sub-sampled training set once the race moves forward.

The resulting RACSAM algorithm consists then in a modified version of the F-race algorithm, described in Section 2.1, where at the qth step only a subsample of the training set of size $\hat{N}^{(q)}$ is used for computing the accuracy of all the M_{q-1} configurations still in race. Equation (1) is then replaced by $\hat{y}_{q,m} = LL(x_q, D_{\hat{N}^{(q)}}, \mathcal{S}_m)$ where $\hat{N}^{(q)}$ is the size of the sub-sampled training set at the qth step. In our preliminary experiments we used the rule $\hat{N}^{(q+1)} = \hat{N}^{(q)} + 10$ with $\hat{N}^{(0)} = 50$ to update the size of the sub-sampled training set.

3 The exploration strategy

The RACSAM algorithm takes as input a set of M candidate feature sets and returns a set of W winners. While this approach makes possible an exhaustive exploration of the feature subspace (i.e $M = 2^n$) in the case of very small n, it requires some modifications if we intend to address problems with very large dimensionality. Our search strategy is quite simple. An initial set $\Theta_0^{(0)}$ of candidates is created either randomly or by adopting some filtering techniques (e.g. Pearson correlation or Gram-Schmidt orthogonalization [7]). This initial set is passed through the RACSAM algorithm which returns the W better candidates. Then an iterative procedure begins. The iterative search is composed of two steps: (i) generation (e.g. by neighborhood exploration) of a new set of M candidates $\Theta_0^{(i)}$ starting from the output $\Theta^{(i-1)}$ of the RACSAM algorithm (ii) racing of $\Theta_0^{(i)}$ by RACSAM. Alternative search strategies that could be easily combined with the RACSAM approach are discussed in [10].

4 Experimental results

Two experiments were carried out: the first one studies the ability of the algorithm to detect the subset of relevant variables in a very large set of irrelevant ones. The second one assesses the improvement in prediction accuracy that can be obtained by adopting the RACSAM approach.

Selection of relevant variables: we consider a problem of feature selection where the dimensionality of the input space is $n >> 10$ and the output is dependent only on 10 inputs according to the relation

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 1/2)^2 + 10x_4 + 5x_5 + \\ + 10\sin(\pi x_6 x_7) + 20(x_8 - 1/2)^2 + 10x_9 + 5x_{10} + \varepsilon.$$

We suppose that the input vector x takes value in the hypercube $[0,1]^n$ and that ε is a standard Gaussian random variable. Note that this feature selection problem is a harder version of the problem proposed in [6]. We conduct two main experiments: the first with $n = 110$ and the second with $n = 210$ inputs. Each experiment has two variants: initialization by Gram Schmidt (GS) and by random generation (RN). For each experiment we generate 25 times a random training and test set by sampling uniformly the input hypercube. In the GS variant we first rank the variables according to the Gram-Schmidt orthogonalization procedure and we create an initial set of 5000 candidates $\Theta_0^{(0)}$ by combining the most relevant 50 variables according to the GS procedure. In the random variant the initial set $\Theta_0^{(0)}$ of 5000 candidates is generated randomly.

The RACSAM procedure performs a F-race where the p-value threshold is set to 0.01 and the number of winners to $W = 50$. The algorithm is stopped when at least 10000 models have been assessed. For each run, we define successful the RACSAM algorithm if it is able to discover and retain the best feature set (made of the first ten variables only) among the W winners. The success rate of the RACSAM algorithm is compared to that of a forward selection (FS) procedure using a LL algorithm with 270 training samples and 500 test samples.

Table 1 reports for each experiment (i) the RACSAM percentage of successes, (ii) the rate of success of the forward selection procedure, (iii) the average number of models (plus the standard deviation) which were examined by the RACSAM algorithm before discovering the correct set, (iv) the average number of training samples (plus the standard deviation) required for the assessment, (v) the average number of test samples (plus the standard deviation) required for the assessment. Note that an exhaustive search would have required the assessment of

n/Init	Racs	FS	Assessed	Train	Test
110/GS	92%	8%	7342 ± 572	259 ± 31	601 ± 307
110/RN	88%	8%	7054 ± 726	220 ± 21	213 ± 214
210/GS	88%	4%	7403 ± 595	265 ± 52	660 ± 522
210/RN	84%	4%	7017 ± 901	223 ± 37	240 ± 366

Table 1. Comparison RACSAM vs. Forward Selection.

2^n models. The experimental results show that the performance of the algorithm is independent of the initialization and quite robust to number of irrelevant variables.

Prediction accuracy: this experiment compares the performance accuracy of the LL algorithm enhanced by the RACSAM procedure to the the accuracy of two state-of-art algorithms, a SVM for regression and a regression tree (RTREE). These two algorithms are well-known and powerful examples of embedded techniques for feature selection. We use the implementations available in the `e1071` and `tree` R packages, respectively. The comparisons were carried out by performing a five-fold cross-validation on six real datasets[3] of high dimensionality: `Ailerons` ($N = 14308, n = 40$), `Pole` ($N = 15000, n = 48$), `Elevators` ($N = 16599, n = 18$), `Triazines` ($N = 186, n = 60$), `Wisconsin` ($N = 194, n = 32$) and `Census` ($N = 22784, n = 137$).

Two version of the RACSAM algorithm were tested: the first (LL-RAC1) takes as feature set the best one (in terms of estimate Mean absolute Error (MAE)) among the W winning candidates : the second (LL-RAC1) averages the predictions of W LL predictors, where each LL model takes as inputs one of the W sets returned by the RACSAM procedure. In both cases we set $W = 5$, and the p-value to 0.01. The selection procedure stops when at least 1000 different configurations have been analysed.

In Table 2 we present, for each learning method, the absolute prediction error averaged over the 5 cross-validation groups. Since the methods are tested on the same examples under the same conditions, we use the paired test of significance to perform an exhaustive paired comparison of all the methods for all the benchmarks. In what follows, by "significantly better" we mean better at least at a 5% significance level. As far as the comparison LL-RAC1 to LL-RAC2 is concerned, we obtained that LL-RAC2 is significantly better than LL-RAC1 3 times out of 6 and it is never significantly worse than LL-RAC1. As far as the comparison of LL-RAC2 to the other state-of-the-art techniques is concerned we obtain that the LL-RAC2 approach is never significantly worse than SVM and/or RTREE but that it performs 5 times out of 6 significantly better than SVM and 6 times out of 6 significantly better than RTREE.

[3] available at http://www.liacc.up.pt/∼ltorgo/Regression/DataSets.html

Dataset	AIL	POL	ELE	TRI	WIS	CEN
LL-RAC1	9.7e-5	3.12	1.6e-3	0.21	27.39	0.17
LL-RAC2	9.0e-5	3.13	1.5e-3	0.12	27.41	0.16
SVM	1.3e-4	26.5	1.9e-3	0.11	29.91	0.21
RTREE	1.8e-4	8.80	3.1e-3	0.11	33.02	0.17

Table 2. Mean Absolute prediction errors.

5 Conclusions

Preliminary results show the effectiveness of the RACSAM approach in selecting relevant features and in improving the predictive accuracy, especially in the case of a combination of the predictors based on the feature sets returned by RACSAM. Future research will extend these preliminary results by combining the RACSAM assessment procedure with more sophisticate search strategies (e.g. the schemata search proposed in [10]) and by applying the technique to massive datasets with thousands of variables.

References

[1] M. Birattari, G. Bontempi, and H. Bersini. Lazy learning meets the recursive least-squares algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *NIPS 11*, pages 375–381, Cambridge, 1999. MIT Press.

[2] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon, editor, *GECCO 2002*, pages 11–18. Morgan Kaufmann, 2002.

[3] G. Bontempi, M. Birattari, and H. Bersini. Lazy learning for modeling and control design. *International Journal of Control*, 72(7/8):643–658, 1999.

[4] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, USA, third edition, 1999.

[5] A. Dean and D. Voss. *Design and Analysis of Experiments*. Springer Verlag, New York, NY, USA, 1999.

[6] J. H. Friedamn. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141, 1991.

[7] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[8] G. H. John and P. Langley. Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining*. AAAI/MIT Press, 1996.

[9] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[10] O. Maron and A. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review*, 11(1–5):193–225, 1997.

[11] Andrew W. Moore and Mary S. Lee. Efficient algorithms for minimizing cross validation error. In *International Conference on Machine Learning*, pages 190–198. Morgan Kaufmann Publishers, Inc., 1994.