ERAF: A R PACKAGE FOR REGRESSION AND FORECASTING

M. Filippone,[1] F. Masulli, [1,2] S. Rovetta[1,3]

[1] *INFM - Istituto Nazionale per la Fisica della Materia*
Via Dodecaneso 33, 16146 Genova, Italy

[2] *Dipartimento di Informatica, Universita' di Pisa*
Via F. Buonarroti 2, 56125 Pisa, Italy
masulli@di.unipi.it

[3] *Dipartimento di Informatica e Scienze dell'Informazione*
Universita' di Genova, Via Dodecaneso 35, 16146 Genova, Italy

Abstract We present a package for R language containing a set of tools for regression using ensembles of learning machines and for time series forecasting. The package contains implementations of Bagging and Adaboost for regression, and algorithms for computing mutual information, autocorrelation and false nearest neighbors.

Keywords: R package, statistical computing, time series, Bagging, Adaboost, Takens-Mane theorem, mutual information, autocorrelation, false nearest neighbors.

Introduction

R is a programming language and environment for statistical computing similar to the S language and environment which was developed at Bell Laboratories [Becker, 1984; Venables and Ripley, 2002; Ihaka and Gentleman, 1996]. It provides a large set of tools optimized for a wide range of problems. It is based on objects such as vectors, matrices, and more complex structures (data frames, lists). There are many operators acting directly on these objects, which make any computation fast and expressed in a straightforward way. These properties, its GNU license [1] and a generic resemblance to Matlab (which shares with R the presence of matrices and vectors as native objects), have boosted its diffusion in the statistical and machine learning communities.

Among the available tools there are packages for multilayer perceptrons, for support vector machines, for multivariate optimization. Moreover, the lan-

guage features all standard programming constructs (conditional instructions, loops) and a very handy 2D and 3D graphics drawing capability.

It is an interpreted language allowing to the user an easy interactive development and usage of programs. The peculiar conventions adopted make it a straightforward task and allow even very complicated constructs to be expressed compactly. The drawback of those properties is a possible penalization of speed. Therefore, it is possible to call external C, C++, or Fortran routines from within an R program. This is useful when parts of the code are computation-intensive and difficult to optimize in R. Moreover, recently a R to C compiler has been released (see `http://hipersoft.cs.rice.edu/rcc/`).

In this paper we present an overview of ERAF, a R package containing a set of algorithms implementing Ensembles for Regression and for time series Analysis and Forecasting we have implemented [2]. In next sections we illustrate some learning machines made available from other R packages (Sect. 1) and the tools included in ERAF: namely some ensemble methods that can be used in regression tasks (Sect. 1) and a set of procedures for time series analysis making possible to transform a forecasting problem in a regression problem (Sect. 2). In Sect. 3 we present a test case. Conclusions are in Sect. 4.

1. Ensembles for regression

Base learners and ensembles

Many base learners are available in R as components of standard packages, including*Multilayer perceptrons* (MLP) and *Support vector machines* (SVM).

Multilayer perceptrons (MLP) are implemented in R through the function *nnet* of the package *nnet* contributed by Venables and Ripley [Ripley, 1996; Venables and Ripley, 2002]. The following parameters can be set: type of architecture (multilayer or single layer perceptron), number of hidden layer units $nnhl$, weight decay parameter λ, stopping criterion (on cost threshold and/or on maximum number of iterations); initialization values for weights, and activation function for output units.

Support vector machines (SVM) [Cortes and Vapnik, 1995] are implemented in R through the function *svm* of package *e1071*. The implementation is the porting of Chang and Lin code [Chang and Lin, 2001; Chang and Lin, 2002]. The adjustable parameters are: kernel type (linear, polynomial, Gaussian), kernel parameters (γ for Gaussian kernel, p and b_0 for polynomial kernel), regularization parameter C, and ε in Vapnik's loss function.

Due the large number of parameters to be set in both *nnet* and *svm*, we included in ERAF package two meta-learners allowing the user to to evaluate the test set error (root mean square), while scanning the parameters, in order to speeding-up the model selection procedure.

The generalization of a learning machine using a finite data set has been studied in the frameworks of the notions of margin [Vapnik, 1998] and of classical bias-variance decomposition of the error [Geman et al., 1992], that recently have been shown to be equivalent [Domingos, 2000].

Ensemble methods [Valentini and Masulli, 2002] aggregate the output of a set of base learners and can increase generalization on the same data set, as they can boost margins, reduce variance, and also bias. The overall effectiveness of a learning machine depends on the specific characteristics of the base learners (more details are, e.g., in [Valentini and Dietterich, 2003]).

In ERAF package we implemented the Bagging [Breiman, 1996] and the Adaboost [Freund and Schapire, 1996] algorithms that are two powerful ensemble methods based on data set re-sampling that have been extensively studied in classification task. The implementations we have enclosed in ERAF package are tailored for regression tasks.

Bagging

Bagging (Bootstrap AGGregatING) [Breiman, 1996] makes a bootstrapping on a dataset consisting in creating new data sets by sampling with replacement from the original data, with equal probabilities for each data item. The basic algorithm creates a model for each new data set and then combining the different estimations thus obtained, by an averaging operation. More formally, starting from the original dataset $\mathcal{L} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$ we build p new training sets $\mathcal{L}_k$ with $k = 1, \cdots, p$ sampling from $\mathcal{L}$ with replacement. A model $f_k(\mathbf{x}, \mathcal{L}_k)$ is identified from each new dataset, then the predictive model is built as

$$f(\mathbf{x}, \mathcal{L}) = \frac{1}{p} \sum_{i=1}^{p} f_k(\mathbf{x}, \mathcal{L}_k) \tag{1}$$

Adaboost

Adaboost [Freund and Schapire, 1996] stands for ADAptive BOOSTing, meaning that the procedure is adaptive with respect to the level of complexity of the training set. The implemented algorithm for regression follows [Drucker, 1997].

The algorithm starts by assigning a probability $p_i^{(1)} = 1/l$ to be sampled to each of the l data items belonging to the set $\mathcal{L} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_l, y_l)\}$. A training set $\mathcal{L}_1$ is generated by sampling with replacement from the original set $l^{(1)}$ examples, and the first learner is trained. We obtain $f^{(1)}$ which gives the output $\hat{y}_i^{(1)}$ for each $\mathbf{x}_i \in \mathcal{L}$. Then we compute the loss $L_i^{(1)}$ selectable among

the following:

$$L_i^{(1)} = \frac{|\hat{y}_i^{(1)} - y_i|}{D}; \qquad L_i^{(1)} = \left(\frac{|\hat{y}_i^{(1)} - y_i|}{D}\right)^2; \qquad L_i^{(1)} = 1 - e^{-\frac{|\hat{y}_i^{(1)} - y_i|}{D}} \tag{2}$$

where D is a normalization constant such that $L_i \in [0, 1]$, i.e.,

$$D = \max_i |\hat{y}_i^{(1)} - y_i|. \tag{3}$$

Then we compute $\bar{L}^{(1)} = \sum_{i=1}^{l^{(1)}} L_i^{(1)} p_i^{(1)}$, i.e., the average of the $L_i^{(1)}$ weighted on $p_i^{(1)}$, and $\beta^{(1)} = \frac{\bar{L}^{(1)}}{1-\bar{L}^{(1)}}$, a quantity whose value is inversely related to the quality of learning as measured on $\mathcal{L}$: Now the sampling probabilities are updated as follows:

$$p_i^{(1)} = p_i^{(1)} \left(\beta^{(1)}\right)^{1-L_i^{(1)}} \tag{4}$$

and of course they are normalized to 1. With this procedure we can assign a larger sampling probability to the examples featuring the larger error. It is iterated until a value of T is reached such that $\bar{L}^{(T)}$ is larger than 0.5 or a selected number of iterations is reached.

The ensemble thus obtained yields a output on a given $\mathbf{x}_i$ which is computed as the median of the $\hat{y}_i^{(t)}$ weighted with the corresponding $\beta^{(t)}$. (The median is used to give robustness to the method.) We consider $\hat{y}_i^{(t)}$ and the corresponding $\beta^{(t)}$ of all T machines which took part to the procedure. They are renamed so that $\hat{y}_i^{(1)} < \hat{y}_i^{(2)} < \cdots < \hat{y}_i^{(T)}$, keeping intact the association between a $\hat{y}_i^{(t)}$ and $\beta^{(t)}$. Then $\log \frac{1}{\beta^{(t)}}$ is summed over t until

$$\sum_t \log \frac{1}{\beta^{(t)}} \geq \frac{1}{2} \sum_t \log \frac{1}{\beta^{(t)}} \tag{5}$$

If t^* is the minimum value of t such that (5) holds, the output $\hat{y}_i$ is that made by machine t^*, that is, $\hat{y}_i = \hat{y}_i^{(t^*)}$.

2. Time series analysis and forecasting

From forecasting to regression

The forecasting problem requires modeling an unknown system, which is assumed to generate the observed time series. Given a time series of n elements $(s_1, s_2, \cdots, s_n)$ obtained by sampling an observed variable of the system, the Takens-Mane [Takens, 1981; Mane, 1981] theorem guarantees that its dynamics can be reconstructed in the space of vectors

$$\mathbf{y}_i = (s_i, s_{i+T}, \cdots, s_{i+(d-1)T}) \tag{6}$$

T and d must be selected appropriately for the dynamics to be correctly reconstructed. Therefore we have implemented the mutual information, autocorrelation, and nearest neighbors algorithms [Abarbanel, 1996] as T can be estimated as the the first minimum of mutual information or as the first zero crossing of autocorrelation, and then we can estimate d using, e.g, false nearest neighbors algorithm [Abarbanel, 1996]. ERAF package includes also two local learners for forecasting, as proposed in [Abarbanel, 1996].

Mutual information

The algorithm for computing *mutual information* implements the following definition [Abarbanel, 1996]:

$$I(T) = \sum_{s_i, s_{i+T}} P(s_i, s_{i+T}) \log_2 \left(\frac{P(s_i, s_{i+T})}{P(s_i) P(s_{i+T})} \right) \tag{7}$$

The interval (a, b) is split into in k contiguous subintervals (typically hundreds) $A = (u, v) = \Delta_1 \cup \Delta_2 \cup \Delta_3 \cup \cdots \cup \Delta_k$, with $\Delta_1 = \left(u, u + \frac{v-u}{k}\right)$, and $\Delta_j = \left[u + (j-1)\frac{v-u}{k}, u + j\frac{v-u}{k}\right)$, $j = 2, \cdots, k$.

To obtaining $P(s_i)$ we count how many s_i belong to each subinterval, then divide by N. $P(s_i)$ is therefore an object of type $\{P_i\}_{i=1,k}$. To obtain the joint probability we build a $k \times k$ matrix J. Element $J_{i,j}$ counts how many pairs (s_i, s_{i+T}) are such that $s_i \in \Delta_i$ and $s_{i+T} \in \Delta_j$i (this quantity is then divided by the number of pairs $N - 1$). Therefore mutual information is computed as:

$$I(T) = \sum_{i=1}^{k} \sum_{j=1}^{k} J_{i,j} \log_2 \left(\frac{J_{i,j}}{P_i P_j} \right) \tag{8}$$

where the summation is limited to $P_i P_j \neq 0$.

Autocorrelation

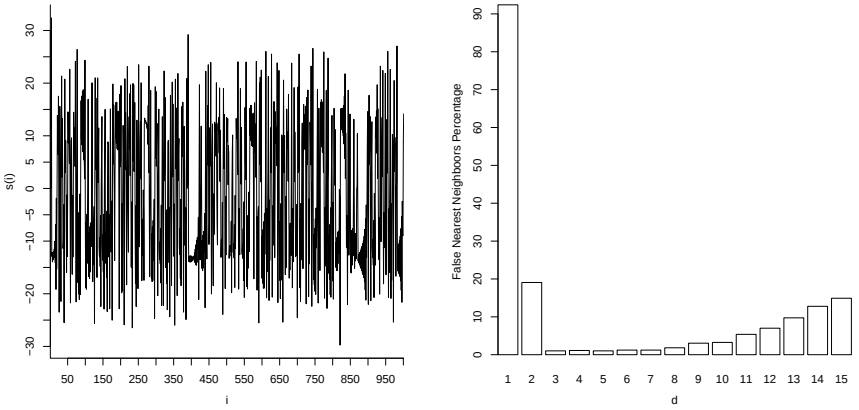
The *autocorrelation* is defined as:

$$C(T) = \sum_i (s_i - \bar{s})(s_{i+T} - \bar{s}) \tag{9}$$

where $\bar{s} = \frac{1}{N} \sum_{i=1}^{N} s_i$ is the average of s [Abarbanel, 1996]. The algorithm is written in C.

False nearest neighbors

The *false nearest neighbors* algorithm [Abarbanel, 1996] allows us to estimate the embedding dimension of a dynamical system. After choosing T corresponding to the first minimum of mutual information or to the first zero crossing of autocorrelation, we consider an element $\mathbf{y}(k) = (s_k, \cdots, s_{k+(d-1)T})$



(a) (b)

Figure 1. (a) The Lorenz time series (b) False nearest neighbors vs embedding dimension

and we search the vector $\mathbf{y}^{NN}(k) = \mathbf{y}(u) = (s_u, \cdots, s_{u+(d-1)T})$ closest to it. To assess whether the vectors $\mathbf{y}(k)$ and nearest neighbor $\mathbf{y}^{NN}(k)$ are close or far in passing from the current space to the space obtained by adding the next coordinate, we check for one of the following two conditions [Abarbanel, 1996]:

$$\frac{|s_{k+dT} - s_{u+dT}|}{\sqrt{\sum_{m=1}^{d} \left(s(k+(m-1)T) - s(u-(m-1)T)\right)^2}} > 15 \qquad (10)$$

$$\frac{|s(k+dT) - s(u+dT)|}{\frac{1}{N}\sum_{k=1}^{N}(s(k) - \bar{s})^2} > 2 \qquad (11)$$

If one of them is fulfilled, we consider $\mathbf{y}^{NN}(k)$ as a false neighbor of $\mathbf{y}(k)$ We repeat the procedure for all vectors $\mathbf{y}(k)$ and compute the percentage of false neighbors. The computation is made starting from $d = 1$ up to a selected maximum value of d. The algorithm for searching the vector $\mathbf{y}^{NN}(k)$ is written in C and optimized following [Nene and Nayar, 1997] and is quite well performing in terms of speed. Selection of T and data processing are made by a procedure written in R.

3. Case study

In this section we present some experimental results concerning the application of ensemble methods on a time series forecasting problem. In particular we compare their performance with those of base learners.
The time series chosen is the well known Lorenz chaotic series obtained sam-

	MLP	SVM	Bagging MLP	Bagging SVM	Adaboost MLP	Adaboost SVM
$rmse$	2.13	2.76	2.42	2.89	**2.02**	3.02

Table 1. Results on the Lorenz time series.

pling the x variable, solution of this system of differential equations [Abarbanel, 1996]:

$$\begin{cases} \dot{x} = \sigma(y - x) & \sigma = 16 \\ \dot{y} = -xz + rx - y & b = 4 \\ \dot{z} = xy - bz & r = 45.92 \end{cases} \tag{12}$$

Using a short time sampling (e.g. $\tau_{\mathrm{s}} = 10^{-2}$) the forecasting problem is easy and all considered learning machines (MLP, SVM and their bagged and boosted versions) obtained similar good generalization results. In the following, we present a numerical experiment where we strongly sub-sampled the series x ($\tau_{\mathrm{s}} = 0.2$) obtaining the series of 1000 values shown in Fig. 1.

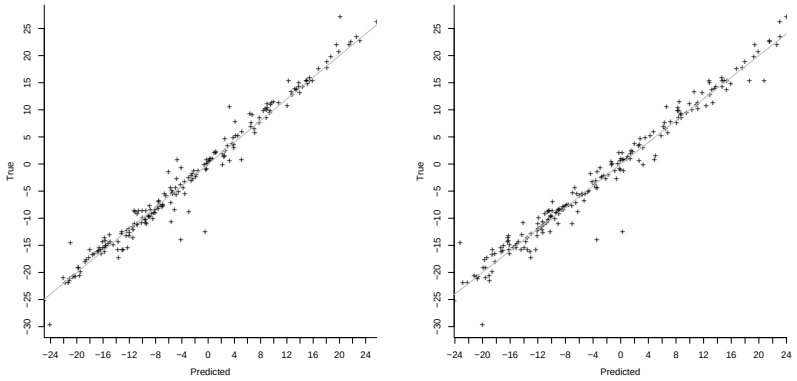
The problem was to forecast the last 200 values using the first 800 for training. The minimum of mutual information was $T = 1$. We used this value in the construction of the time-delayed coordinates vectors. Be applying the false nearest neighbors algorithm to those vectors we obtained $d = 4$ as an estimate of the embedding dimension.

Using this value of d we trained many MLP and SVM in order to find the best set of parameters that leads to the minimum of the root mean square error ($rmse$) on the test set. All the necessary software is made available by the ERAF package.

Then we built ensemble methods with base learners with the best set of parameters. All the ensembles were made up by 100 base learners using training sets of the same dimension of the original training set. The best set of parameters for this problem were $\gamma = 5$, $C = 5$ and $\varepsilon = 0.005$ for SVM using Gaussian kernel and $nnhl = 36$ and $\lambda = 0.1$ for MLP (see Sect. 1). All the results are shown in Table 1. In Fig. 2 we can see the differences between the regression lines for the best MLP and Adaboost with MLP.

4. Conclusions

The procedures contained in the ERAF package we have described in this paper allow the R programmer to face regression and time series forecasting problems using state of the art methods. In particular, ERAF makes available:



(a) (b)

Figure 2. (a) Regression line for Adaboost with MLP (b) Regression line for MLP

- Bagging and Adaboost meta-learners that can improve the generalization results in regression tasks of base learners, such as multi-layer perceptron and support vector machines, already available in R;

- Tools for calculating mutual information, autocorrelation and false nearest neighbors allowing the user to turn a forecasting problem into a regression problem, on the basis of the embedding theorem and related prescriptions.

We are extending the ERAF package in order to make available local forecasters [Abarbanel, 1996], and the procedures for Singular-Spectrum Analysis [Vautard and Ghil, 1989; Vautard et al., 1992] able to extract trends from a time series.

Acknowledgments

Work funded by the Italian National Institute for the Physics of Matter (INFM), and the Italian Ministry of Education, University and Research (MIUR).

Notes

1. The R language is available at `http://www.r-project.org/` for the most common computer platforms (Windows, Linux, Mac OS).
2. The package is available at `http://mlsc.disi.unige.it/R`.

References

H.D.I. Abarbanel. *Analysis of Observed Chaotic Data*. Springer, New York, 1996.

R.A. Becker and J.M. Chambers. *Design of the S System for Data Analysis*. Comm. A.C.M., 27:5 pp. 486-495, 1984.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

C. C. Chang and C J. Lin. *Training ν-support vector classifiers: Theory and algorithms*. Neural Computation, 13(9):2119-2147, 2001.

C. C. Chang and C J. Lin. *Training ν-support vector regression: Theory and algorithms*. Neural Computation, 14(8):1959-1977, 2002.

C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20:273–297, 1995.

P. Domingos. A Unifi ed Bias-Variance Decomposition for Zero-One and Squared Loss In Proceedings of the Seventeenth National Conference on Artifi cial Intelligence, pages 564-569, Austin, TX, AAAI Press, 2000.

H. Drucker. Improving regressors using boosting techniques. In D. H. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 107–115. Morgan Kaufmann, 1997.

Y. Freund and R.E. Schapire. *Experiments with a New Boosting Algorithm.*. Proceedings of the Thirteenth Conference, ed: L. Saitta, Morgan Kaufmann, pp. 148-156, 1996.

S. Geman, E. Bienenstock, and R. Doursat. *Neural networks and the bias-variance dilemma.*. Neural Computation, 4(1):1-58, 1992.

R. Ihaka and R. Gentleman. *R: A language for data analysis and graphics*. Journal of Computational and Graphical Statistics, 5(3):299-314, 1996.

R. Mane. On the dimension of the compact invariant sets of certain non-linear maps. In D.A Rand and L. S. Young, editors, *Dynamical Systems and Turbulence*, *Lecture Notes in Mathematics*, vol. 898 p. 230–242, 1981. Springer-Verlag, Berlin.

S.A. Nene and S.K. Nayar. *A simple algorithm for nearest neighbor search in high dimensions*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19:989-1003, 1997.

B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge, 1996.

F. Takens. Detecting strange attractors in turbulence. In D.A. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence*, *Lecture Notes in Mathematics*, vol. 898, pp. 366–381, 1981. Springer-Verlag, Berlin.

R. Vautard and M. Ghil. Singular-spectrum analysis in nonlinear dynamics, with applications to paleoclimatic time series. *Physica D*, 35:395–424, 1989.

R. Vautard, P. You, and M. Ghil. Singular-spectrum analysis: A toolkit for short, noisy chaotic signals. *Physica D*, 58:95–126, 1992.

G. Valentini and F. Masulli. Ensembles of Learning Machines, in M. Marinaro and R. Tagliaferri, editors, Neural Nets WIRN Vietri-02, Series Lecture Notes in Computer Sciences, Springer-Verlag, Heidelberg (Germany), vol. 2486, pp.3-19, 2002

G. Valentini and T. G. Dietterich. Bias-variance analysis of Support Vector Machines for the development of SVM-based ensemble methods Journal of Machine Learning Research (in press).

V. N. Vapnik. *Statistical Learning Theory* . Wiley, New York, 1998.

W.N. Venables and B.D. Ripley. *Modern Applied Statistics with S*. Springer, 2002.