Lazy Learners at work: the Lazy Learning Toolbox

Gianluca Bontempi, Mauro Birattari, Hugues Bersini

Iridia - CP 194/6
Université Libre de Bruxelles
50, Av. Franklin Roosvelt
1050 Bruxelles, Belgium
Phone:+32-2-6503168; Fax:+32-2-6502715
E-mail: {gbonte,mbiro,bersini}@ulb.ac.be

Abstract

Lazy Learning is a memory-based technique that, once a query is received, extracts a prediction interpolating locally the neighboring examples of the query which are considered relevant according to a distance measure. In previous works, we presented a Lazy Learning method which selects automatically on a query-by-query basis the optimal number of neighbors to be considered for each prediction. This learning method proved to be a very effective technique in a number of academic and industrial case studies, ranging from time series prediction to data modeling and nonlinear control.

This paper discusses the implementation of the Lazy Learning technique in a toolbox for use with Matlab© and its successful application to the problem of multivariate regression proposed by the Third Erudit Competition.

1 Introduction

Lazy learning (Aha, 1997) is a local learning technique which postpones all the computation until an explicit request for a prediction is received. The request is fulfilled interpolating locally the examples considered relevant according to a distance measure. Each prediction requires therefore a local modeling procedure that can be seen as composed of a *structural* and of a *parametric* identification. The parametric identification consists in the optimization of the parameters of the local approximator. The structural identification involves, among other things, the selection of a family of local approximators, the selection of a metric to evaluate which examples are more relevant, and the selection of the *bandwidth* which indicates the size of the region in which the data are correctly modeled by members of the chosen family of approximators. For a comprehensive tutorial on local learning and for further references see Atkeson *et al.* (1997).

In previous published works (Bontempi *et al.*, 1998; Bontempi *et al.*, 1999a; Birattari *et al.*, 1999), the authors proposed a method in which the family of local approximators and the bandwidth are selected locally and tailored for each query point by means of a local leave-one-out cross-validation (Stone, 1974). The problem of bandwidth selection is then reduced to the selection of the number k of neighboring examples which are given a non-zero weight in the local modeling procedure.

The technique has been tested on a series of difficult modeling and control problem, including the participation to the International Competition on Time Series in Leuven (Suykens & Vandewalle, 1998), where the Lazy technique ranked second and the participation to the Third International Erudit competition[1] where the Lazy technique was awarded as a runner-up. This technique is now implemented in the Lazy Learning Toolbox for use with Matlab© publicly available on the Web[2].

The *Lazy Learning Toolbox for use with Matlab*© (Birattari & Bontempi, 1999) implements a Lazy Learning technique, which considers polynomials of different degrees and allows a local model selection, as well as a local combination of approximators of different degrees. In particular, each time a prediction is required for a specific query point, a set of local models is identified, each with a different polynomial degree and each including a

[1] `http://www.erudit.de/erudit/activities/ic-99/index.htm`
[2] `http://iridia.ulb.ac.be/~lazy/`

different number of neighbors. The generalization ability of each model is then assessed through a local leave-one-out. Finally, a prediction is obtained either combining or selecting the different models on the basis of some statistic of their cross-validation errors.

The major feature of this toolbox consists in the adoption of the *recursive least squares* algorithm for the identification of the local models (Birattari *et al.*, 1999). This is an appealing and efficient solution to the intrinsically incremental problem of identifying and validating a sequence of local linear models centered in the query point, each including a growing number of neighbors. It is worth noticing here that a leave-one-out cross-validation of each model considered does not involve any significant computational overload, since it is obtained though the PRESS statistic (Myers, 1994) which simply uses partial results returned by the recursive least squares algorithm.

The outline of the paper is as follows. In Section 2 we will introduce the Lazy Learning method in formal notation. The set of functions for input/output data modeling provided by the toolbox are discussed in Section 3. A number of successful applications of the Lazy Learning technique to academic and industrial problems are presented in Section 4. The experimental application of the Lazy Learning toolbox to the prediction for the 3rd Eufit competition is discussed in Section 4.1. The large variety of application examples should give the reader a clear and detailed picture of the strength of the methodology. Section concludes the paper by illustrating the advantages of a Lazy approach to data driven tasks.

2 The Lazy Learning method

Given two variables $\mathbf{x} \in \Re^m$ and $y \in \Re$, let us consider the mapping f: $\Re^m \to \Re$, known only through a set of n examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ obtained as follows:

$$y_i = f(\mathbf{x}_i) + \varepsilon_i, \tag{1}$$

where $\forall i$, ε_i is a random variable such that $E[\varepsilon_i] = 0$ and $E[\varepsilon_i \varepsilon_j] = 0$, $\forall j \neq i$, and such that $E[\varepsilon_i^r] = \mu_r(\mathbf{x}_i)$, $\forall r \geq 2$, where $\mu_r(\cdot)$ is the unknown r^{th} moment of the distribution of ε_i and is defined as a function of $\mathbf{x}_i$.

Given a query point $\mathbf{x}_q$, the parameter $\boldsymbol{\beta}_1$ of a local first-degree polynomial approximating $f(\cdot)$ in a neighborhood of $\mathbf{x}_q$, can be obtained solving the local polynomial regression:

$$\sum_{i=1}^{n} \left\{ (y_i - \mathbf{x}_i' \boldsymbol{\beta})^2 K\left(\frac{D(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \right\}, \tag{2}$$

where, given a metric on the space $\Re^m$, $D(\mathbf{x}_i, \mathbf{x}_q)$ is the distance from the query point to the i^{th} example, $K(\cdot)$ is a weight function, h is the bandwidth, and where the vectors $\mathbf{x}_i$ have been obtained by pre-appending a constant value 1 to each vector $\mathbf{x}_i$ in order to consider a constant term in the regression.

Once obtained the local first-degree polynomial approximation, a prediction of $y_q = f(\mathbf{x}_q)$, is finally given by:

$$\hat{y}_q = \mathbf{x}_q' \hat{\boldsymbol{\beta}} \tag{3}$$

By exploiting the linearity in the parameters of the local approximator, a leave-one-out cross-validation estimation of the error variance $E[(y_q - \hat{y}_q)^2]$ can be obtained without any significant overload. In fact, using the PRESS statistic (Myers, 1994), it is possible to calculate the leave-one-out error $e_j^{cv} = y_j - \mathbf{x}_j' \hat{\boldsymbol{\beta}}^{-j}$, without explicitly identifying the regression parameters $\hat{\boldsymbol{\beta}}^{-j}$ with the j^{th} case set aside.

If a rectangular weight function $K(\cdot)$ is adopted, the optimization of the parameter h can be conveniently reduced to the optimization of the number k of neighbors to which a unitary weight is assigned in the local regression evaluation. In other words, we reduce the problem of bandwidth selection to a search in the space of $h(k) = D(\mathbf{x}(k), \mathbf{x}_q)$, where $\mathbf{x}(k)$ is the k^{th} nearest neighbor of the query point.

The main advantage deriving from the adoption of an indicator weight function, is that, simply by updating the parameter $\hat{\boldsymbol{\beta}}(k)$ of the model identified using the k nearest neighbors, it is straightforward and inexpensive to obtain $\hat{\boldsymbol{\beta}}(k+1)$. The recursive algorithm described in (Birattari *et al.*, 1999; Bontempi *et al.*, 1999d) returns for a given query point $\mathbf{x}_q$, a set of predictions $\hat{y}_q(k) = \mathbf{x}_q' \hat{\boldsymbol{\beta}}(k)$, together with a set of associated leave-one-out error vectors $\mathbf{e}^{cv}(k)$.

On the basis of this information a final prediction $\hat{y}_q$ of the value of the regression function can be obtained in two different ways: the first is based on the selection of the *best* approximator according to a given criterion, while the second returns a prediction as a *combination* of more local models.

If the selection paradigm, frequently called *winner-takes-all*, is adopted, the most natural way to extract a final prediction $\hat{y}_{d,q}$, consists in comparing, on the basis of the classical *mean square error* criterion, the prediction obtained for each value of k, given the degree d of the local approximator.

$$\hat{y}_q = \mathbf{x}_q' \hat{\boldsymbol{\beta}}(\hat{k}), \quad \text{with } \hat{k} = \arg\min_{k \in \mathcal{K}} mse^{cv}(k); \tag{4}$$

where $\mathcal{K}$ is a range, defined by the analyst, from which the optimal number of neighbors is selected.

As an alternative to the *winner-takes-all* paradigm, the toolbox proposes the local combinations of estimates (Wolpert, 1992). In this case the final prediction of the value y_q is obtained as a weighted average of the best b models, where b is a parameter of the algorithm. Suppose the predictions $\hat{y}_q(k)$ and the error vectors $\mathbf{e}^{cv}(k)$ have been ordered creating a sequence of integers $\{k_i\}$ so that $mse^{cv}(k_i) \leq mse^{cv}(k_j)$, $\forall i < j$. The prediction of y_q is given by

$$\hat{y}_{d,q} = \frac{\sum_{i=1}^{b} \zeta_i \, \hat{y}_q(k_i)}{\sum_{i=1}^{b} \zeta_i}, \tag{5}$$

where the weights are the inverse of the mean square errors: $\zeta_i = 1/mse^{cv}(k_i)$. This is an example of the *generalized ensemble method* (Perrone & Cooper, 1993).

3 The toolbox

The toolbox is a collection of Matlab© functions, optimized through `mex` subroutines, which provides the designer with a set of Lazy Learning techniques for regression input/output problems. The user is required to provide a training input/output set, a test input set and to choose between these Lazy Learning approximators

Winner-takes-all Lazy Learning with constant models. For each query, this method identifies and validates, using a growing number of nearest neighbor, a number of different local polynomial approximators of degree 0. Among these models, the best one is selected by Eq. (4).

Winner-takes-all Lazy Learning with linear models. For each query, this method identifies and validates using a growing number of nearest-neighbors, a number of different local polynomial approximators of degree 1. Among these models, the best one is selected by Eq. (4).

Winner-takes-all Lazy Learning with quadratic models. For each query, this method identifies and validates using a growing number of nearest-neighbors, a number of different local polynomial approximators of degree 2. Among these models, the best one is selected by Eq. (4).

Lazy Learning with local combination. For each query, this method identifies and validates using a growing number of nearest-neighbors, a number of different local polynomial approximators of degree 0, 1, and 2. According to the value of some input variables, the function selects the best one as in Eq. (4), or combines a number of best models as in Eq. (5).

4 Applications

The Lazy Learning method described in the above sections is currently applied with success by the Iridia laboratory to a number of academic and industrial problems. Here we list some of them:

Financial prediction of stock markets. This is a joint project of Iridia and the research center of Masterfood, which adopts the Lazy Learning techniques for the prediction of some market indices.

Prediction of chaotic time series. The Lazy Learning method (Bontempi *et al.*, 1999b) ranked second among 17 participants to the International Competition on Time Series organized by the *International Workshop on Advanced Black-box techniques for nonlinear modeling* in Leuven, Belgium (Suykens & Vandewalle, 1998).

Non linear control and identification task. The Lazy Learning method is used to implement adaptive control strategies based on the extension of linear control techniques to the nonlinear setting (Bontempi *et al.*, 1999a; Bontempi *et al.*, 1999c). The method is currently tested on a set of benchmarks proposed in the Esprit project FAMIMO (Fuzzy Algorithm for Multi Input Multi Output process).

Modeling of industrial processes. The Lazy Learning technique is employed to model the rolling steel mill process of the FaFer Usinor steel company in Charleroi, Belgium. It is also the subject of an active collaboration of Iridia with the Honeywell Technology Center in Minneapolis, U.S..

Electric load forecasting. A joint project of Iridia and Tractebel Belgium is adopting local Lazy techniques for the forecasting of the electrical load.

Prediction of economic variables. A joint project of Iridia and Dieteren, the first Belgian car dealer, is studying the adoption of Lazy Learning techniques to predict the annual amount of sales on the basis of historical data.

4.1 The competition task

In order to compare the Lazy method with state-of-the-art and innovative soft computing technologies we decided to take part to the Third International Erudit Competition "Protecting rivers and streams by monitoring chemical concentrations and algae communities".

The competition is based on the observations collected during a biological research on the state of pollution of European rivers. The aim of the research was to identify the chemical control variables relevant for the biological processes. During the research study water quality samples were taken from sites on different European rivers of a period of approximately one year. These samples were analyzed for various chemical substances.

The relationship between the chemical and biological features is known to be quite complex. Typical of such real-life problems, the particular data set for the problem contains a mixture of (fuzzy) qualitative variables and numerical measurement values, with much of the data being incomplete. The competition task is the prediction of algae frequency distributions on the basis of the measured concentrations of the chemical substances and the global information concerning the season when the sample was taken, the river size and its flow velocity. The two last variables are given as linguistic variables.

An amount of 340 samples were taken, each containing 18 features. The first 11 features are the season, the river size, the fluid velocity and 8 chemical concentrations which should be relevant for the algae population distribution. The last 7 values of each data set are the distribution of different kinds of algae and represent the outputs to be predicted. Each entrant in the competition received 200 complete data sets (training data) and 140 data sets (evaluation data) containing only the 11 values of the river descriptions and the chemical concentrations.

4.2 The Lazy Learning procedure

This is the sequence of steps we adopted to model the multi-input multi-output regression task:

Missing data treatment. Since the software has been conceived to deal with complete and real training sets, the first operation was to replace the qualitative values with numerical indices and the missing values with the averages of the known attributes. Then, the multi-input multi-output problem was decomposed in a series of multi-input single-output tasks.

Definition of a validation procedure. In order to validate the different approaches and models we decided to adopt a leave-one-out technique, due to the low number of training cases.

Modeling with state-of-the-art techniques. We tested on the same conditions both a conventional linear multivariable regression and *Cubist*, a commercial regression tree developed by Ross Quinlan for generating piecewise-linear models[3].

Modeling with Lazy Learning. The leave-one-out procedure was used to define the most appropriate range of neighbors $\mathcal{K}$ in Eq. (4) and the number b in Eq. (5). Once tuned these parameters, the combined Lazy Learning techniques appeared to outperform both the linear model and the Cubist regression tree. This result confirmed the good performances obtained on multivariable regression tasks in (Birattari *et al.*, 1999).

Final prediction. The Lazy Learning approximator was chosen as the approximator expected to perform the best on a generalization set. Another nice feature of the Lazy approach was the capacity to adapt to the missing feature in the test without requiring neither the replacement of the missing attributes nor the retraining of ad-hoc models.

[3]For more details on Cubist see `http://www.rulequest.com`

After a final check of the conformity between the distribution of the predicted values and the distribution of the output values in the training set, the prediction set was submitted. The whole procedure required about one week of design time.

5 Why is it convenient to be lazy?

The amount of practical applications and the good results in two competitions make of Lazy Learning a promising tool for modeling and prediction.

In this conclusive section we summarize the main advantages of a Lazy Learning approach:

Reuse of linear techniques in parametric identification. The local weighted regression in (2) is an example of fast parametric techniques which returns in a single step the best set of local parameters.

Reuse of linear techniques in structural identification and model validation. The PRESS statistic is a well-founded and effective way to assess a local model in cross-validation.

Fast design. Local learning methods are often made target of criticism as far as the prediction time is concerned. What the detractors forget, is that a fair comparison between learning methods should also account for the design time. For example, the design time for the data analysis problem of the Erudit competition should have been prohibitive in the case of a generic nonlinear method, like a neural network. Let us consider the time required to select the correct number of neurons or layers for a neural network that approximates a multi-input multi-output mapping with 11 inputs and 7 outputs. On the contrary, model selection is done automatically in a Lazy Learning method with the advantage that different configurations can be tested and assessed at a faster rate.

Reuse of linear control techniques in nonlinear control. The paradigm of local learning enables the reuse of well-founded techniques of linear control (e.g. pole placement, linear optimal control) in a nonlinear setting (Bontempi *et al.*, 1999a).

Reuse of linear forecasting techniques in nonlinear prediction. The paradigm of local learning enables the reuse of well-founded techniques for linear forecasting (e.g. AR models) in nonlinear forecasting tasks (Bontempi *et al.*, 1999b).

Intrinsically adaptive. Lazy Learning is intrinsically adaptive since the learning procedure remains unchanged with the update of the training set. No retraining or incremental learning is required (Bontempi *et al.*, 1999c).

No negative interference. Lazy Learning is practically insensitive to the phenomenon of negative interference, which occurs in global modeling techniques when new training samples in previously uncovered regions of the input space are observed (Atkeson *et al.*, 1997).

Acknowledgments

The work of Gianluca Bontempi was supported by the European Union TMR Grant FMBICT960692. The work of Mauro Birattari was supported by the FIRST program of the Région Wallonne, Belgium.

References

Aha D. W. 1997. Editorial. *Artificial Intelligence Review*, **11**(1–5), 1–6.

Atkeson C. G. , Moore A. W. & Schaal S. 1997. Locally weighted learning. *Artificial Intelligence Review*, **11**(1–5), 11–73.

Birattari M. & Bontempi G. . 1999. *The Lazy Learning Toolbox, For use with Matlab.* Tech. rept. TR/IRIDIA/99-7. IRIDIA-ULB, Brussels, Belgium.

Birattari M. , Bontempi G. & Bersini H. 1999. Lazy learning meets the recursive least-squares algorithm. *In:* Kearns M. S. , Solla S. A. & Cohn D. A. (eds), *Advances in Neural Information Processing Systems 11.* Cambridge: MIT Press.

Bontempi G. , Birattari M. & Bersini H. . 1998. Recursive lazy learning for modeling and control. *Pages 292–303 of: Machine Learning: ECML-98 (10th European Conference on Machine Learning)*.

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

Bontempi G. , Birattari M. & Bersini H. 1999b. Local learning for iterated time-series prediction. *Pages 32–38 of:* Bratko I. & Dzeroski S. (eds), *Machine Learning: Proceedings of the Sixteenth International Conference*. San Francisco, CA: Morgan Kaufmann Publishers.

Bontempi G. , Bersini H. & Birattari M. 1999c. The local paradigm for modeling and control: From neuro-fuzzy to lazy learning. *Fuzzy Sets and Systems*. in press.

Bontempi G. , Birattari M. & Bersini H. 1999d. A model selection approach for local learning. *Artificial Intelligence Communications*. in press.

Myers R. H. 1994. *Classical and Modern Regression with Applications*. second edition edn. Boston, MA: PWS-KENT Publishing Company.

Perrone M. P. & Cooper L. N. 1993. When networks disagree: Ensemble methods for hybrid neural networks. *Pages 126–142 of:* Mammone R. J. (ed), *Artificial Neural Networks for Speech and Vision*. Chapman and Hall.

Stone M. 1974. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, **36**(1), 111–147.

Suykens J. A. K. & Vandewalle J. (eds) 1998. *Nonlinear Modeling: Advanced Black-Box Techniques*. Kluwer Academic Publishers. Chap. The K. U. Leuven Time Series Prediction Competition, pages 241–251.

Wolpert D. 1992. Stacked Generalization. *Neural Networks*, **5**, 241–259.