Lazy Learning Vs. Speedy Gonzales: A fast algorithm for recursive identification and recursive validation of local constant models

Mauro Birattari and Gianluca Bontempi

IRIDIA
Université Libre de Bruxelles
Brussels, Belgium
{*mbiro, gbonte*} *@ulb.ac.be*

Abstract

In this paper we propose a recursive method for identifying and cross-validating local constant models. The algorithm we derive here is intended to be a part of a more general *lazy learning* method already presented by the authors (Birattari *et al.*, 1999). We take for granted aspects related to the search of the nearest-neighbors, the definition of a metric and local combination of estimators, and we focus our attention on the derivation of an efficient way to obtain and assess a sequence of local constant models centered on a given query point, and each including a growing number of nearest-neighbors.

1 Introduction

In this paper we deal with the classical problem of *supervised learning.* A set of examples $\{(x_i, y_i)\}_{i=1}^{n}$ is given, where $\forall i\ x_i \in \Re^m$ and $y_i \in \Re$. From this dataset, we wish to *learn* the real mapping $f\colon \Re^m \mapsto \Re$ which is implicitly supposed to have somewhat informed the generation of the examples. In particular we want to be able to *generalize*, i.e. to predict the value y_q which would be associated as an *output* to a given *input* vector $x_q \in \Re^m$, even if the original dataset does not explicitly include any example about x_q.

A first possible way to tackle the above stated problem consists in fitting a *global* parametric model to the available examples, and then evaluating this model in x_q in order to obtain a prediction $\hat{y}_q$ of the scalar y_q. For a generic dataset, it is usually necessary to resort to a complex universal approximator as for instance to a neural network, which happens to be nonlinear in its parameter and therefore to be quite expensive to tune and to validate.

In this paper we refer to an alternative approach, known as *lazy learning* (Aha, 1997), which belongs to the family of *local* approaches (Bottou &

Vapnik, 1992). In order to obtain a prediction of y_q, a lazy learning method selects a local training set among the neighbors of the query point x_q, and uses this local information to extract the value $\hat{y}_q$.

On a local scale, the restriction to polynomial approximators is widely accepted and is not considered as a sensible limitation to the expressiveness of a learning method. Furthermore, the adoption of a local polynomial approximator yields major computational advantages deriving from the fact that a polynomial model is linear in its parameters and can be thus tuned and assessed through efficient and well-understood methods from linear statistics. For a comprehensive tutorial on local learning and for further references see Atkeson *et al.* (1997).

In a previous paper (Birattari *et al.*, 1999), we have already proposed a general lazy learning framework in which, for each query point, a prediction is obtained by combining (Wolpert, 1992), on the basis of a local leave-one-out cross-validation, a number of local approximators of different degree, and each identified using a different number of neighbors. For each of the considered polynomial degrees, a recursive least squares algorithm is used to identify a sequence of local models centered in the query point, each including a growing number of neighbors. The leave-one-out cross-validation of each of these models does not involve a significant computational overload, since it is obtained though the PRESS statistic (Myers, 1994) which, in our implementation, uses partial results returned by the recursive identification algorithm. The method described above, can be used to recursively identify and validate local models of any degree and then, in principle, also for constant models i.e. for polynomials of degree zero.

Anyway a far more efficient implementation is possible which fully exploits properties peculiar to constant models. In this paper we focus on the derivation of such an algorithm for recursive identification and recursive leave-one-out validation of local polynomial approximators of degree zero. We take for granted that an appropriate metric has been defined in the input space $\Re^m$, that a rectangular weighting kernel has been adopted, and that an algorithm has been chosen to (efficiently) retrieve from the original dataset the K-nearest-neighbors of a given query point x_q. We assume also that it is valuable to obtain a sequence of prediction yielded by constant models, each identified on the basis of a growing number of nearest-neighbors of the query point, together with their respective mean square error in cross-validation. In other words, we suppose that a method has been defined in order to extract a final prediction starting from a sequence of approximators of degree 0, and from their leave-one-out assessment, and in case from equivalent sequences of higher degree approximators identified and validated through an appropriate algorithm (Birattari *et al.*, 1999).

2 Local constant models and local assessment

We suppose that a subset of K nearest-neighbors of the query point x_q at hand has been selected. The sequence $\{(x_i, y_i)\}_{i=1}^{K}$ will be, from here on, the sequence of the K-nearest-neighbors ordered so that $\varrho(x_i, x_q) \leq \varrho(x_j, x_q)$, $\forall i \leq j$, where

ϱ is an appropriate distance function in the space $\Re^m$.

A generic local constant model identified on the first k nearest neighbors is the classical *sample average* (Papoulis, 1991) of the outputs associated to the nearest k examples:

$$\hat{y}(k) = \frac{1}{k}\sum_{i=1}^{k} y_i = \hat{\mu}(k). \tag{1}$$

A leave-one-out mean square error of this model is obtain as follows:

$$mse^{cv}(k) = \frac{1}{k}\sum_{j=1}^{k}\left(\varepsilon_j^{cv}(k)\right)^2, \tag{2}$$

where $\varepsilon_j^{cv}(k)$ is the error in the prediction of the j^{th} neighbor, yielded by the model identified on the k nearest-neighbors with the j^{th} removed:

$$\begin{aligned}
\varepsilon_j^{cv}(k) &= y_j - \hat{y}_{-j}(k) = y_j - \frac{\sum_{\substack{i=1\\ i\neq j}}^{k} y_i}{k-1} \\
&= y_j - \frac{\sum_{i=1}^{k} y_i}{k-1} = y_j - \frac{k\frac{\sum_{i=1}^{k} y_i}{k} - y_j}{k-1} \\
&= y_j - \frac{k\hat{\mu}(k) - y_j}{k-1} = \frac{ky_j - k\hat{\mu}(k)}{k-1} \\
&= \frac{k}{k-1}\left(y_j - \hat{\mu}(k)\right) = \frac{k}{k-1}\left(y_j - \hat{y}(k)\right) \\
&= \frac{k}{k-1}\varepsilon_j(k).
\end{aligned} \tag{3}$$

Eq. 3 shows that the leave-one-out error for the j^{th} neighbor is a linear function of the re-substitution error and does not depend on x_j. From Eq. 2 and 3, it follows that:

$$\begin{aligned}
mse^{cv}(k) &= \frac{\sum_{j=1}^{k}\left(\frac{k}{k-1}\varepsilon_j(k)\right)^2}{k} = \frac{\frac{k^2}{(k-1)^2}\sum_{j=1}^{k}\left(y_j - \hat{\mu}(k)\right)^2}{k} \\
&= \frac{k}{k-1}\frac{\sum_{j=1}^{k}\left(y_j - \hat{\mu}(k)\right)^2}{k-1} = \frac{k}{k-1}\hat{\sigma}^2(k),
\end{aligned} \tag{4}$$

where $\hat{\sigma}^2(k)$ is the *sample variance* (Papoulis, 1991) of the output associated to the nearest k examples.

3 The recursive algorithm

In Sec. 2 we have defined the local prediction and the leave-one-out mean square error obtained from the first k nearest-neighbors for a generic value of k.

In this section we will derive a recursive formulation of Eq. 1 and 4, i.e. we will make explicit the equations that allow the computation of $\hat{y}(k)$ and $mse^{cv}(k)$ starting from $\hat{y}(k-1)$, $mse^{cv}(k-1)$, and the k^{th} nearest-neighbor y_k.

From the recursive formulation of the average $\hat{\mu}(k)$ and of the variance $\hat{\sigma}^2(k)$ (see appendix):

$$\hat{\mu}(k) = \frac{k-1}{k}\hat{\mu}(k-1) + \frac{1}{k}y_k, \tag{5}$$

and

$$\hat{\sigma}^2(k) = \frac{k-2}{k-1}\hat{\sigma}^2(k-1) + \frac{1}{k}\big(y_k - \hat{\mu}(k-1)\big)^2, \tag{6}$$

and from the results of Eq. 1 and 4 we obtain the recursive formulation of the prediction:

$$\hat{y}(k) = \frac{k-1}{k}\hat{y}(k-1) + \frac{1}{k}y_k, \tag{7}$$

and the recursive formulation of the leave-one-out mean square error:

$$mse^{cv}(k) = \frac{k(k-2)^2}{(k-1)^3}mse^{cv}(k-1) + \frac{1}{k-1}\big(y_k - \hat{\mu}(k-1)\big)^2. \tag{8}$$

The algorithm described by Eq. 7 and 8 computes, for a given query, the sequence of the predictions and the sequence of the mean square errors when a growing number of nearest-neighbors is used as local training sub-set.

The recursion in Eq. 7 is initialized for $k = 1$ with $\hat{y}(1) = y_1$, i.e. with the output associated with the nearest-neighbor. On the contrary, the recursion on the mean square error is started for $k = 2$ since a leave-one-out error cannot be defined for less than two examples. Furthermore, it is worth notice here, that $mse^{cv}(1)$ does not need to be explicitly initialized since for $k = 2$ the first term in Eq. 8 equals zero because of the numerator of its coefficient.

4 Discussion

In figure 1 we propose a comparison between the recursive algorithm described by Eq. 7 and 8 and its non-recursive counterpart obtained from the direct implementation of Eq. 1 and 2. For a given query and once the neighbors have been retrieved, the plot shows the time[1] needed by the two methods in order to fit and assess all the constant models which consider a number of neighbors in the range between 2 and K, for values of K between 3 and 50. Figure 1 visually confirms that the time needed by the recursive algorithm grows linearly with K, as it could be expected from the nature of Eq. 7 and 8.

[1]The experiments were performed on a Pentium 400MHz CPU.

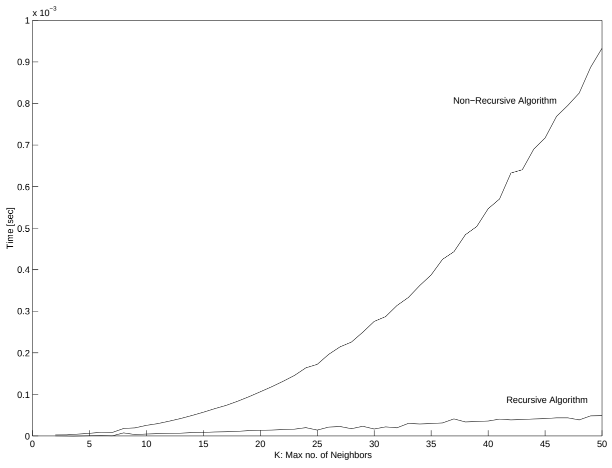

Figure 1: Time of computation needed to fit and asses all the models which consider a number of neighbors in the range between 2 and K.

A comparison between the algorithm developed in Sec. 3 and the *lazy* recursive least square (Birattari *et al.*, 1999) for the identification of linear models is of interest here.

The recursive least square, as implemented by Birattari *et al.* (1999), does not return for every value of k the "exact" model that would be obtained by solving off-line the corresponding least square problem on the first k neighbors. The returned model is rather the model that would be obtained by solving off-line a *ridge regression* (Draper & Smith, 1981). This implicit effect of regularization is obtained through the conventional initialization of the variance/covariance matrix (Bierman, 1977), and prevents problems due to a nearly singular local data matrix.

Local constant models do not suffer from this kind of problems: the single parameter that needs to be identified is, for a given query and for a given value of k, a function only of the output y_i and not of the input x_i of the k-nearest-neighbors. Therefore, the position of the nearest-neighbors in the input space is not relevant, and it is not necessary to adopt any regularization method. From this, it follows that the predictions return by Eq. 7 and the mean square errors returned by Eq. 8 are "exact" i.e. are identical to the values that would be obtained by their off-line counterparts.

Appendix

We derive here the results used in Sec. 3. The recursive formulation of the sample average (5) can be obtained as follows:

$$\begin{aligned}\hat{\mu}(k) &= \frac{1}{k}\sum_{i=1}^{k} y_i = \frac{\sum_{i=1}^{k-1} y_i + y_k}{k} \\ &= \frac{(k-1)\frac{\sum_{i=1}^{k-1} y_i}{k-1} + y_k}{k} = \frac{(k-1)\hat{\mu}(k-1) + y_k}{k} \\ &= \frac{k-1}{k}\hat{\mu}(k-1) + \frac{1}{k} y_k.\end{aligned} \tag{9}$$

The basis of the recursive computation of the sample variance (6) can be obtained as follows:

$$\begin{aligned}\hat{\sigma}^2(k) &= \frac{\sum_{j=1}^{k}\big(y_j - \hat{\mu}(k)\big)^2}{k-1} \\ &= \frac{1}{k-1}\sum_{j=1}^{k}\left(y_j - \frac{(k-1)\hat{\mu}(k-1) + y_k}{k}\right)^2 \\ &= \frac{1}{k-1}\sum_{j=1}^{k}\left(y_j - \frac{k\hat{\mu}(k-1) - \hat{\mu}(k-1) + y_k}{k}\right)^2 \\ &= \frac{1}{k-1}\sum_{j=1}^{k}\left(\big(y_j - \hat{\mu}(k-1)\big) - \frac{y_k - \hat{\mu}(k-1)}{k}\right)^2 \\ &= \frac{1}{k-1}\sum_{j=1}^{k}\Bigg(\big(y_j - \hat{\mu}(k-1)\big)^2 + \left(\frac{y_k - \hat{\mu}(k-1)}{k}\right)^2 \\ &\qquad - 2\big(y_j - \hat{\mu}(k-1)\big)\frac{y_k - \hat{\mu}(k-1)}{k}\Bigg) \\ &= \frac{1}{k-1}\sum_{j=1}^{k}\big(y_j - \hat{\mu}(k-1)\big)^2 + \frac{1}{k-1}\sum_{j=1}^{k}\left(\frac{y_k - \hat{\mu}(k-1)}{k}\right)^2 \\ &\qquad - 2\frac{1}{k-1}\sum_{j=1}^{k}\left(\big(y_j - \hat{\mu}(k-1)\big)\frac{\big(y_k - \hat{\mu}(k-1)\big)}{k}\right) \\ &= \frac{1}{k-1}\left(\sum_{j=1}^{k-1}\big(y_j - \hat{\mu}(k-1)\big)^2 + \big(y_k - \hat{\mu}(k-1)\big)^2\right) \\ &\qquad + \frac{1}{k-1}k\left(\frac{y_k - \hat{\mu}(k-1)}{k}\right)^2\end{aligned}$$

$$
\begin{aligned}
&- \frac{2}{k-1} \frac{\left(y_k - \hat{\mu}(k-1)\right)}{k} \left(\sum_{j=1}^{k} y_j - k\hat{\mu}(k-1) \right) \\
&= \frac{k-2}{k-1} \frac{\sum_{j=1}^{k-1} \left(y_j - \hat{\mu}(k-1)\right)^2}{k-2} + \frac{1}{k-1} \left(y_k - \hat{\mu}(k-1)\right)^2 \\
&\quad + \frac{1}{k(k-1)} \left(y_k - \hat{\mu}(k-1)\right)^2 \\
&\quad - \frac{2}{k(k-1)} \left(y_k - \hat{\mu}(k-1)\right) \left(\sum_{j=1}^{k-1} y_j + y_k - k\hat{\mu}(k-1) \right) \\
&= \frac{k-2}{k-1} \hat{\sigma}^2(k-1) + \frac{k+1}{k(k-1)} \left(y_k - \hat{\mu}(k-1)\right)^2 \\
&\quad - \frac{2}{k(k-1)} \left(y_k - \hat{\mu}(k-1)\right) \left((k-1)\hat{\mu}(k-1) + y_k - k\hat{\mu}(k-1) \right) \\
&= \frac{k-2}{k-1} \hat{\sigma}^2(k-1) + \frac{k+1}{k(k-1)} \left(y_k - \hat{\mu}(k-1)\right)^2 \\
&\quad - \frac{2}{k(k-1)} \left(y_k - \hat{\mu}(k-1)\right)^2 \\
&= \frac{k-2}{k-1} \hat{\sigma}^2(k-1) + \frac{1}{k} \left(y_k - \hat{\mu}(k-1)\right)^2.
\end{aligned}
$$

Acknowledgments

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

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.

Bierman G. J. 1977. *Factorization Methods for Discrete Sequential Estimation.* New York, NY: Academic Press.

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.

Bottou L. & Vapnik V. N. 1992. Local learning algorithms. *Neural Computation*, **4**(6), 888–900.

Draper N. R. & Smith H. 1981. *Applied Regression Analysis.* New York: John Wiley and Sons.

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

Papoulis A. 1991. *Probability, Random Variables, and Stochastic Processes.* Third edn. Electrical & Electronic Engineering Series. New York, NY: McGraw-Hill International Editions.

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