Geometrical Selection of Important Inputs with Feedforward Neural Network[0]

Fabrice ROSSI[1]

THOMSON-CSF/AIRSYS/RD/RDTE
7-9, rue des Mathurins
92221 BAGNEUX, FRANCE

Université Paris-IX Dauphine
UFR MD
Place du Maréchal de Lattre de Tassigny
75016 PARIS, FRANCE

e-mail: rossi@ceremade.dauphine.fr

Abstract

In this paper, we introduce a method that allows to evaluate efficiently the "importance" of each coordinate of the input vector of a neural network. This measurement can be used to obtain informations about the studied data. It can also be used to suppress irrelevant inputs in order to speed up the classification process conducted by the network.

1 Introduction

Variable selection is one of the key issues of classification tasks (and more generally of model estimation). In order to solve a classification problem, we start in general with a lot of measures coming from the real world. If these measurements are well chosen and if the problem is simple enough, we can reasonably assume that a classifier could be designed from the data. Unfortunately, it will in general perform its classification as a black-box, without giving any information about the underlying task. One of the direct application of variable selection is to suppress the useless input variables of the classifier: this suppression helps to understand the classification and give important informations about the task itself. Moreover, the suppression allows to simplify the classifier itself and therefore to speed up the classification process. It is also very important to notice that variable suppression speeds up the acquisition process (i.e., the measurement of the real world in order to produce the input variables).

A huge work has been done by statisticians in order to study variable selection when the regression tool is linear (e.g., [6]). The purpose of this article is not to review these methods but to work on problems that cannot be correctly solved with linear tools. When the classifier is non linear, the problem is more complex and several neural based method have been proposed to solve it (e.g., [2, 8, 9]). In this paper, we propose an extension of a previously introduced method [9] and compare it with other neural approaches and with a statistical method which was proposed in the neural network community [1].

The remainder of this paper is organized as follows. Section 2 introduces the mathematical aspect of our variable selection method and compares it to existing methods. Section 3 gives some experimental results on artificial and real-world data.

2 Geometrical Variable Selection (GVS)

2.1 The proposed method

The method presented in this section allows to choose which attributes to suppress in an efficient and reliable way. The key idea is to analyze an already trained classifier in order to measure how much its calculation depends on each attribute: the evaluation is global. Moreover, the method can be applied to non linear classifiers and therefore solve the linearity limitation of traditional methods.

Let $F(x, w)$ be a parametric classifying function (also called a non-linear regression model, e.g. a MLP): x is an input vector, w is a weight vector that allows to modify the computation performed by F (e.g. the connection weights in a MLP) and $F(x, w)$ is the output of the function which belongs to $[0, 1]^c$, where c is the number of classes of the problem. Let C_k be the studied k-th class and let $\chi_{C_k}(x)$ be the membership function corresponding to C_k, i.e. $\chi_{C_k}(x) = 1 \Leftrightarrow x \in C_k$. Let $C(x)$ be the perfect classifying function, i.e.

[0]This work was performed on Mrs Kim K. PHAM's responsibility, at THOMSON-CSF/AIRSYS.
Published in ICANNGA'97 Proceedings.
Available at
http://apiacoa.org/publications/1997/icannaga97.pdf

[1]Up to date contact informations for Fabrice Rossi are avaible at http://apiacoa.org/

$C(x) = (\chi_{C_1}(x), \chi_{C_2}(x), \ldots, \chi_{C_c}(x))$. The goal of the learning phase is to find w such that $F(x,w) \simeq C(x)$. In general w is chosen in order to minimize some distance criterion between $F(x,w)$ and $C(x)$ (for instance, the total quadratic error, $E(w) = \sum_{x \in T} \|F(x,w) - C(x)\|^2$, where T is the training set of the classification task). In fact, minimizing this quadratic distance is equivalent to minimizing a quadratic probabilistic distance between $F(x,w)_k$ (the k-th output of F) and $P(C_k \mid x)$, the *a posteriori* probability of class C_k, given x (see [10]).

Let us assume that we can compute the differential of F with respect to its first variable (i.e., the input vector), called $\frac{\partial F}{\partial x}(x,w)$. The optimal Bayes decision rule [3], if F_k is a good approximation of $P(C_k \mid x)$, is to assume that x belongs to class j if $F_j(x,w)$ is strictly greater than $F_i(x,w)$ for all $i \neq j$. The boundary between C_k and C_l is the set of point $x \in \partial C^{k,l}$ such that $F_k(x,w) = F_l(x,w)$ and for all $i \neq k$ and $i \neq l$, $F_i(x,w) < F_l(x,w)$. Therefore, the boundary is locally described by the equation $F_k(x,w) = F_l(x,w)$. Then, the unitary normal to the boundary at point x is $n^{k,l}(x)$, given by:

$$n^{k,l}(x) = \frac{\frac{\partial F_k(x,w)}{\partial x} - \frac{\partial F_l(x,w)}{\partial x}}{\left\| \frac{\partial F_k(x,w)}{\partial x} - \frac{\partial F_l(x,w)}{\partial x} \right\|} \tag{1}$$

A low value for $\left|n_i^{k,l}(x)\right|$ (the i-th coordinate of the normal unitary vector) shows that the boundary normal is perpendicular to the i-th coordinate axis and therefore that the local separation hyperplane (which approximate the boundary) contains this axis: the i-th coordinate of the input vector is locally useless[2] for separating elements from C_k from elements of C_l. Therefore, $\left|n_i^{k,l}(x)\right|$ is a good measure of the local importance of the i-th coordinate axis.

The remaining problem is to combine the $\left|n_i^{k,l}(x)\right|$ when x belongs to $\partial C^{k,l}$ in order to obtain a global understanding of the relative importance of the different coordinate axis. On a theoretical point of view, it will be interesting to compute the mean normal vector of boundary between C_k and other classes. But C_k may contain several separated clusters for which the boundaries are parallel but with opposite normal vectors: therefore, computing the integral of $n^{k,l}$ on $\partial C^{k,l}$ is not really meaningful.

In order to obtain a simple criterion, we define the following matrix:

$$S_{k,i} = \sum_{l \neq k} \int_{\partial C^{k,l}} \left|n_i^{k,l}(x)\right| d\sigma_x, \tag{2}$$

The main problem is now to obtain points belonging to $\partial C^{k,l}$. As this boundary has at most a dimension of $n-1$ (if n is the dimension of the input space of the classifier, i.e., the number of variables), we cannot obtain points in $\partial C^{k,l}$ with a random selection. In order to find a point belonging to $\partial C^{k,l}$, with start by randomly choosing two points (in the training or validation set), x_1 and x_2 such that $x_1$is classified by F in C_k and x_2 is classified in C_l. Then, we study the function

$$g(\lambda) = F_k\left(\lambda x_1 + (1-\lambda)x_2\right) - F_l\left(\lambda x_1 + (1-\lambda)x_2\right) \tag{3}$$

The definition of g and assumptions on x_1 and x_2 implies that $g(0) < 0$ and $g(1) > 0$. Moreover, as F is assume to be differentiable with respect to x, g is obviously continuous. Therefore, for a specific $\lambda \in]0,1[$ (which can be easily found with simple zero finding algorithms [7] such as dichotomy), we have $g(\lambda) = 0$. Of course, there is not guaranty that the corresponding $x = \lambda x_1 + (1-\lambda)x_2$ belongs to $\partial C^{k,l}$ (because we can have for a specific j, $F_j(x) > F_k(x)$) but this is quite likely to happen. This method can be used to build a first description of $\partial C^{k,l}$. Of course, this description is far from being perfect, but it is easy to test if the obtained points really belongs to the boundary and therefore, we will end up with $\partial \tilde{C}^{k,l}$, a finite subset of $\partial C^{k,l}$.

The obtained value $S_{k,i}$ is therefore the global score associated to the i-th coordinate axis as a classifying axis for class C_k. Finally, we can define a mean score, the vector S as:

$$S_i = \frac{1}{c} \sum_{k=1}^{c} S_{k,i} \tag{4}$$

S_i is the mean global score associated to the i-th coordinate axis as a classifying axis.

Each coordinate axis of the input space is associated to a variable. Therefore, a high axis score is equivalent to an important variable: in order to suppress variables, we just have to discard the one with the lowest score.

2.2 Links with previous works

The method explained in the previous section is closely related to an algorithm introduced in [8]. In

[2] In fact, this is true only if the tangent hyperplane does not run through the boundary, a case which is quite unlikely to happen.

this article, the authors introduce an attribute ranking method based on first order differentials. There are two important differences between the method presented here and their algorithm:

- Priddy and al. combine the individual differential $\left|\frac{\partial F_k}{\partial x_i}(x,w)\right|$ without normalization ;
- they take into account every examples (and even additional points which are not examples) without focusing on boundary examples.

In fact, the main justification of Priddy and al. is a statistical one, whereas we are working on geometrical arguments, which are in our opinion more suited to the attribute suppression goal.

In a previous paper [9], we have introduced a first version of the method presented in the previous section. This paper shows that this method was more efficient than the one introduced in [8]. The main difference between the old version and the improved one is the use in the current method of an exact determination of boundary examples and an exact value for the normal vector. The current method has therefore stronger theoretical justification. On the real data given in the following section, there is no important differences between results obtained by the previous method and the current one, but such differences were observed for artificial problems (for which GVS performs better).

2.3 Feed-forward neural network case

We have demonstrated in a previous paper [5, 4], that an extended back-propagation algorithm can be defined for arbitrary feed-forward neural networks (including in the same framework MLP, RBF networks and Wavelet Networks [11] for instance). This algorithm allows to compute efficiently the differential of $F(x,w)$ with respect to its input x, $\frac{\partial F}{\partial x}(x,w)$, if F is the output of a neural network (with w as generalized weight vector). Therefore, our algorithm can be applied to any feed-forward neural network.

3 Experiment on real data

Some experiments were conducted on real world data: we have chosen to work with radar data. In this case, we have 32 inputs corresponding to different physical measurements. The goal is to decide whether a given input vector represents a target or some clutter. We have a big database, with around 40 000 points. This database contains only 5 000 target points, therefore, we will characterize the performances with two numbers: the average classification rate on the whole test set and the mean of the recognition rate of both classes. In order to applied a stopped training method, the database was split into three parts: around 20 000 training examples, 10 000 validation examples and 10 000 test examples.

The goal of our simulation is to keep as few variable as possible.

We tried several different methods:

3.1 Neural methods

We trained a simple MLP with 32 inputs, 16 hidden neurons and 2 output neurons (i.e., a 32-16-2 MLP). After 200 iterations of Polak Ribiere Conjugate Gradient [7] (PRCG), the best MLP (selected with the help of the validation set) obtain 96.43 % as classification rate on the test set (and 89.93 % as mean classification rate).

We applied on this best MLP two neural based variable saliency computation methods: OCD [2] and GVS proposed in this article. The saliencies obtained by these algorithms allow to rank the different variables. The order for the 7 best variables is given in the following table:

OCD	2	21	22	18	20	15	6
GVS	2	21	20	22	13	6	15

In fact, there is only one difference: OCD chooses attribute 18, whereas GVS chooses 13.

3.2 Statistical Method

We have also used Battiti's algorithm, as described in [1]. This method combines two measures: the mutual information between a feature and class information, $MI(f, \mathcal{C})$, and the mutual information between a feature and the already selected features, $MI(f_1, f_2, \ldots, f_p, f)$. The algorithm uses a parameter λ which measures the importance of the between features mutual information. It has to be heuristically chosen. Battiti says that values ranging from 0.5 to 1.0 give good results. We have chosen to compare three values: 0.5, 0.75 and 1.0. These choices give the following attribute orders:

λ =0.5	2	22	21	15	14	20	18
λ =0.75	2	32	19	26	6	8	29
λ =1.0	2	32	19	26	6	8	19

The first set ($\lambda = 0.5$) is called B1 and the second one (valid for $\lambda = 0.75$ and $\lambda = 1$) is called B2.

3.2.1 Comparison

In order to compare the different methods (and in fact the selected attribute sets), we have trained a 7-16-2 MLP on the data. For neural based methods, we have pruned in the best MLP the useless inputs and we have retrained the obtained MLP from this starting point. For Battiti's method, we have chosen randomly a starting point for a 7-16-2 MLP. In order to allow a fair comparison, we have increased the learning time for this MLP (400 iterations instead of 200). The performance for one starting point is the classification rate on the test set for the best MLP obtained after the training (the MLP is selected with the help of its mean square error on the validation set). The following table shows the performances:

attribute set	rate	mean rate
GVS	95.81%	87.64%
OCD	95.42 %	87.79%
B1	94.00%	81.70%
B2	93.41%	80.98%

This table shows that our neural network based selection obtains good performances. It overcomes limitations of the mutual information based method [1] which does not select attributes needed to maintain a satisfactory recognition rate of the target examples. Our method obtains results very similar to OCD [2]. The main advantage of our method is that it can be applied to any neural network, even one for which a zero weight does not mean that this weight can be suppressed (for instance RBF networks). Moreover, OCD is quite difficult to use because it cannot be applied if the network is not at a minimum of the error function. This is not the case of GVS.

4 Conclusion

In this paper we have introduced a new method that allows to suppress data attributes in a classification task. The goal of this suppression is to reduce the preprocessing and classification times. It is based on an analysis of the calculation performed by a parametric classifier such as a multi-layer perceptron. With the help of previous results, we have shown that this method was easy to use for arbitrary feed-forward neural networks. An Experiment conducted on real data show that this method is efficient and can therefore be used for real world applications. Additional work is needed to prove the consistency of this method and to demonstrate its performances on other real world data.

References

[1] Roberto Battiti. Using Mutual Information for Selecting Features in Supervised Neural Net Learning. *IEEE Trans. On Neural Networks*, 5(4):537–550, July 1994.

[2] Tautvydas Cibas, Françoise Fogelman Soulié, Patrick Gallinari, and Sarunas Raudys. Variable selection with neural networks. *Neurocomputing*, 8(12):223–248, 1996.

[3] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[4] Cédric Gégout, Bernard Girau, and Fabrice Rossi. A General Feed-Forward Neural Network Model. Technical report NC-TR-95-041, NeuroCOLT, Royal Holloway, University of London, May 1995. Available at `http://apiacoa.org/publications/1995/neurocolt1995.pdf`.

[5] Cédric Gégout, Bernard Girau, and Fabrice Rossi. Generic Back-Propagation in Arbitrary Feedforward Neural Networks. In D. W. Pearson, N. C. Steele, and R. F. Albrecht, editors, *Int. Conf. on Artificial Neural Nets and Genetic Algorithms*, pages 168–171, Alès, April 1995. Springer Verlag.

[6] A. J. Miller. *Subset Selection in Regression.* Chapman and Hall, 1990.

[7] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C.* Cambridge University Press, second edition, 1992.

[8] Kevin L. Priddy, Steven K. Rogers, Dennis W. Ruck, Gregory L. Tarr, and Matthew Kabrisky. Bayesian selection of important features for feedforward neural networks. *Neurocomputing*, 5:91–103, 1993.

[9] Fabrice Rossi. Attribute suppression with multi-layer perceptron. In *CESA Multiconference*, volume Symposium on Robotics and Cybernetics, pages 542–547, Lille-France, July 1996. IMACS.

[10] Halbert White. Learning in Artificial Neural Networks: A Statistical Perspective. *Neural Computation*, 1(4):425–464, 1989.

[11] Qinghua Zhang and Albert Benveniste. Wavelet networks. *IEEE Trans. On Neural Networks*, 3(6):889–898, November 1992.