Int. J. Appl. Math. Comput. Sci., 2007, Vol. 17, No. 1, 107–112
DOI: 10.2478/v10006-007-0010-7

DATA PROBES, VERTICAL TRAJECTORIES AND CLASSIFICATION: A TENTATIVE STUDY

DAVID W. PEARSON

Hubert Curien Laboratory – UMR CNRS 5516
Jean Monnet University of Saint-Etienne
18 rue Benoit Lauras, 42023 Saint-Etienne, France
e-mail: david.pearson@univ-st-etienne.fr

In this paper we introduce a method of classification based on data probes. Data points are considered as point masses in space and a probe is simply a particle that is launched into the space. As the probe passes by data clusters, its trajectory will be influenced by the point masses. We use this information to help us to find vertical trajectories. These are trajectories in the input space that are mapped onto the same value in the output space and correspond to the data classes.

Keywords: classification, output zeroing, nonlinear control, differential geometry

1. Introduction

In this paper we consider a problem related to classification. In supervised learning we usually have access to data pairs (x_k, y_k), for $k = 1, \dots, m$, where the x_k's are observed input vectors and the y_k's are observed response or output vectors. In our particular study, we require that $x_k \in \mathbb{R}^n$ and $y_k \in \mathbb{R}$, although the method could be extended to include $y_k \in \mathbb{R}^p$ but this is not considered in this paper. One of the objectives of supervised learning is to find a mapping $\pi : \mathbb{R}^n \to \mathbb{R}$ that satisfies $\pi(x_k) = y_k$ for $k = 1, \dots, m$. The mapping π could be anything from a simple linear regression model to more sophisticated methods like, e.g., support vector machines or neural networks; the particular choice of the model is of secondary importance in this paper. The relation to classification is that if two input vectors x_i, x_j are deemed to be in the same class and the classes are defined by the output values, then we will have $\pi(x_i) = \pi(x_j) = y_0$ for some particular value y_0. Or, what is more reasonable, $\pi(x_i) \approx \pi(x_j) \approx y_0$ because we are working with real data and the necessary tolerances. If $\pi(x_i) = \pi(x_j) = y_0$ and we can calculate a trajectory in $\mathbb{R}^n$, denoted by $x(t)$ for $0 \le t \le \delta$, such that

- $x(0) = x_i$,
- $x(\delta) = x_j$,
- $\pi(x(t)) = y_0$ for $0 \le t \le \delta$,

then $x(t)$ is called a vertical trajectory.

Our aim is to develop a method of calculating such vertical trajectories from an existing database, i.e., a set of observed data pairs (x_k, y_k), for $k = 1, \dots, m$. Our hypothesis is that these vertical trajectories could be used along with the original data when we wish to identify the parameters of the particular model that we have chosen for the mapping π. We hope that constraining the calculated π to satisfy the vertical trajectory criteria will lead to a better representation of the underlying geometrical structure of the data. Although we are not yet able to prove our hypothesis, our preliminary investigations have produced some promising results. Work along the same lines has been carried out by some researchers, cf. (Benton and Hand, 2002; Hand, Li and Adams, 2001), although ours is purely deterministic and does not take into consideration any notions of probability.

Because we need to generate dynamic data (vertical trajectories) from static data (the (x_k, y_k) pairs), we need to find a method that allows us to do this. In the course of our research we learned that the most satisfying way of doing this was to be found in a field called data sonification. In data sonification, sound is generated from data. The sound can be listened to and interpreted, thus providing an extra dimension to the data and aiding the analysis task. One particular method of generating sound is called 'model based' and essentially generates trajectories based on static data. This is the method presented in (Hermann and Ritter, 1999) and the method that we have adopted.

The differential equations relating to this method are Newtonian and refer to the trajectory of a point mass. It is this point mass that we call a data probe.

We have needed to combine different disciplines together in order to achieve our objectives. Our principal sources of inspiration are nonlinear control theory (Isidori, 1995), differential geometry and, in particular, vertical fields (Hermann, 1964), data sonification (Hermann and Ritter, 1999) and cluster analysis (Chiu, 1994).

We have been investigating vertical fields and their applications for a number of years. In particular, we have been able to look at the structure of neural networks (Pearson, 1996) and, by relating vertical fields to the notion of homogeneous data, we have applied them to pollution forecasting (Pearson and Batton-Hubert, 2005).

The paper is organised into three main sections. The mathematical formulation of probe dynamics and vertical fields are presented in the first section. The subsequent section is devoted to the exploitation of the generated trajectories. Then some results based on simulated data are presented. The work is very much ongoing and so the paper concludes with some perspectives.

2. Mathematical Formulation

As was stated in the introduction, the data are organised into input/output pairs (x_k, y_k) for $k = 1, \dots, m$, where $x_k \in \mathbb{R}^n$ and $y_k \in \mathbb{R}$. For a vector x we denote its components by $x^T = [x^1, \dots, x^n]$ and we use subscripts to indicate different vectors such as x_i and x_j.

2.1. Probe Dynamics. We assume that the data are grouped together into homogeneous sets referred to as clusters. The criterion of homogeneity is that the output value, say y_0, is the same for all data pairs belonging to the same cluster. We also assume that there exists some underlying mapping $\pi : \mathbb{R}^n \to \mathbb{R}$ that satisfies $\pi(x_k) = y_k$ for $k = 1, \dots, m$. For each data pair (x_k, y_k) we define the corresponding data point d_k as

$$d_k = \begin{bmatrix} y_k \\ x_k \end{bmatrix}.$$

The dynamics of the probe follow the model of (Hermann and Ritter, 1999). We begin by defining the following function:

$$\phi(y, x) = \sum_{k=1}^{m} \exp\Big(-\alpha \Big\| \begin{bmatrix} y \\ x \end{bmatrix} - d_k \Big\|^2 \Big), \tag{1}$$

where α is a positive scalar parameter.

A probe is simply a particle that is launched into the data space and influenced by the data points considered as masses. The standard Newtonian equation of motion of the particle contains therefore acceleration, velocity and resistance terms. In order to recover an ordinary first-order autonomous differential equation, we apply the standard trick of augmenting the state space dimension by defining

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} y \\ x \\ \dot{y} \\ \dot{x} \end{bmatrix}, \tag{2}$$

where $\dot{y} = \mathrm{d}y/\mathrm{d}t$ etc. Using (1) and (2), the equation of motion of the probe is

$$\dot{z} = \begin{bmatrix} \dot{z_1} \\ \dot{z_2} \\ \begin{bmatrix} \dot{z_3} \\ \dot{z_4} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} z_3 \\ z_4 \\ \nabla\phi(z_1, z_2) - \gamma \begin{bmatrix} z_3 \\ z_4 \end{bmatrix} \end{bmatrix}, \tag{3}$$

where γ is a positive scalar parameter and

$$\nabla\phi(z_1, z_2) = \begin{bmatrix} \dfrac{\partial\phi}{\partial z^1} \\ \vdots \\ \dfrac{\partial\phi}{\partial z^{n+1}} \end{bmatrix}.$$

Note that the partial derivatives are up to $\partial/\partial z^{n+1}$ because, by definition, $x^n = z^{n+1}$. We note also that this equation is slightly different than the model presented in (Hermann and Ritter, 1999) because we have not included any masses. We assume that all data points have unit mass.

2.2. Vertical Trajectories. Let $\pi(x_0) = y_0$ and let $x(t)$ with $0 \le t \le \delta$ be a trajectory that satisfies $x(0) = x_0$. If the condition $\pi(x(t)) = y_0 \ \forall t \in [0, \delta]$ is satisfied, then $x(t)$ is said to be a vertical trajectory (Hermann, 1964). If the dynamic equation for x is $\dot{x} = f(x)$ for some vector field f, then $\dot{y} = \pi(x)_* f(x)$, where $\pi(x)_*$ is the differential of π. If the trajectory is vertical, then the vector field satisfies $\pi(x)_* f(x) = 0$. This condition will be important in what follows.

In the case of control theory, one use of vertical fields is the following (Isidori, 1995): Assume that we have a system described by the dynamics $\dot{x} = f(x) + ug(x)$ with the output $y = h(x)$, where f and g are vector fields satisfying certain technical conditions (usually smooth or analytic), h is a function (also usually smooth or analytic) and u is a scalar input variable. The objective is to calculate u to satisfy $h(x(t)) = 0$ for $0 \le t \le \delta$. If this is the case, then $f + ug$ will be a vertical field for the function h. This process is called output zeroing.

Vertical trajectories are of interest to us because we are working with supervised learning. Therefore we know the desired output for any given input in the data set. Thus, if the input x_i is associated with the output y_0, then we will have $\pi(x_i) = y_0$ and, if x_j belongs to the same cluster as x_i, then we will also have $\pi(x_j) = y_0$. In other words, we could construct a trajectory $x(t)$ for $0 \leq t \leq \delta$ such that $x(0) = x_i$, $x(\delta) = x_j$ and $\pi(x(t)) = y_0$ for $0 \leq t \leq \delta$, a vertical trajectory linking x_i to x_j. As has been mentioned in the introduction, we cannot really hope to have $\pi(x_i) = \pi(x_j)$ but more like $\pi(x_i) \approx \pi(x_j)$, and so numerical tolerances have to be taken into consideration when implementing the algorithm.

Now consider a data probe. If the probe's trajectory takes it near a cluster, then the value of $z_1(t)$ will be close to some particular output value, say y_0, for some t, by the definition of $z(t)$ (2). If, at the same time as $z_1(t) \approx y_0$, we also have $z_3(t) \approx 0$, then, from the previous paragraphs in this subsection, the trajectory is close to a vertical trajectory at this point. As the probe equations are integrated, we have a sequence of points and each point on a data probe trajectory that satisfies these criteria is kept in memory for further use. We will refer to these points as w_i.

For each point w_i, we can construct a vertical trajectory passing through the point by applying the output zeroing technique as follows (Isidori, 1995): The system equations are based on (2), but with an added feedback term in the z_3 coordinate that zeros the coordinate, i.e., zeros $\dot{y}$. Then the overall system is as follows, where we use v as a variable to indicate that the trajectory is wholly vertical:

$$\begin{bmatrix} \dot{v_1} \\ \dot{v_2} \\ \begin{bmatrix} \dot{v_3} \\ \dot{v_4} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} v_3 \\ v_4 \\ \left[\nabla\phi(v_1, v_2) - \gamma \begin{bmatrix} v_3 \\ v_4 \end{bmatrix} + \begin{bmatrix} g(v) \\ 0 \end{bmatrix} \right] \end{bmatrix} \tag{4}$$

with the system output

$$h(v) = v_1 - y_0, \tag{5}$$

where $g(v)$ is the feedback function to be calculated and $v(0) = w_i$. Following the method presented in (Isidori, 1995), we know that by construction $v_1(0) = y_0$ and, to achieve $h(v(t)) = 0$ for sufficiently small t, it can be shown that $g(v)$ must be set to

$$g(v) = -\frac{\partial\phi(y_0, v_2)}{\partial v_1}.$$

In this way, a set of vertical trajectories can be generated from the data and used in the estimation of the mapping π.

3. Using Vertical Trajectories

Vertical trajectories such as (4) give us information on how the data are arranged in space. We now want to exploit that piece of information in order to calculate the function π. There are, obviously, various ways in which we can do that and, in fact, this will be the subject of future investigations: For the moment we have adopted the following approach. For each centre we consider the trajectories associated with the centre, by looking for the initial points and seeing on which trajectory of Type (2) they lie. Each centre k is then associated with a function π_k as follows:

$$\pi_k(x) = \exp(-\alpha_k \|A_k x - c_k\|^2), \tag{6}$$

where $\alpha_k > 0$ is a parameter fixed in advance (we will come back to this point later), c_k is the centre k and A_k is an $n \times n$ matrix to be calculated. The parameter α_k corresponds to the radius of influence of the centre k and does not necessarily take the same value of α in (1); each centre could have a different value. In reality, we have to integrate all the differential equations numerically and so a vertical trajectory consists of a sequence of vectors v where, from (4), we see that for each vector in the sequence we have symbolically $v_1 = y$, $v_2 = x$, $v_3 = \dot{y}$ and $v_4 = \dot{x}$. Therefore, if the vector v in the sequence is vertical, then we must have

$$\frac{\mathrm{d}\pi}{\mathrm{d}t}(x(t)) = \pi_\star(x(t))\dot{x}(t) = \pi_\star(v_2)v_4 = 0.$$

This provides us with a minimisation criterion to allow us to calculate A_k. We need to minimise the following expression:

$$E_k = \sum_{v \in \text{vertical}_k} (\pi_\star(v_2)v_4)^2,$$

where vertical$_k$ is the set of all sequences of vectors associated with vertical trajectories of the centre k. We would also like the maximum of the function (6) to be achieved when $x = c_k$ and so we add the constraint $A_k c_k = c_k$ to the above to obtain the following constrained optimisation problem:

$$\min \sum_{v \in \text{vertical}_k} (\pi_\star(v_2)v_4)^2 \tag{7a}$$

subject to

$$A_k c_k = c_k. \tag{7b}$$

Each A_k is thus calculated via a constrained minimisation problem (7a) and then an overall function for π is determined as follows: We consider all vertical trajectories associated with a centre and calculate the mean value of all of the v_1 components. For centre k, we refer to this mean value as ω^k. The overall function for π is then given by

$$\pi(x) = \sum_{k=1}^{p} \omega^k \pi_k(x), \tag{8}$$

where p is the number of centres.

4. Some Simulation Results

In this section we illustrate our algorithm using some simulated data. We use such data because, obviously, the geometrical properties are known in advance. Also, the small dimensions of the data allow a clear visual presentation.

The data were randomly generated to belong to two well separated clusters, i.e.,

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} + 0.5r$$

with the output $1 + 0.2s$, and

$$\begin{bmatrix} -2 \\ -1 \end{bmatrix} + 0.5r$$

with the output $1.5+0.2s$, where r is a 2-dimensional vector with elements in $N(0,1)$ and s is a scalar in $N(0,1)$. Twenty data points were generated for each cluster to act as training data. We also generated further ten points for each cluster to act as parameter optimisation data and ten points for each cluster for model validation purposes. Some example data can be seen in Fig. 1.

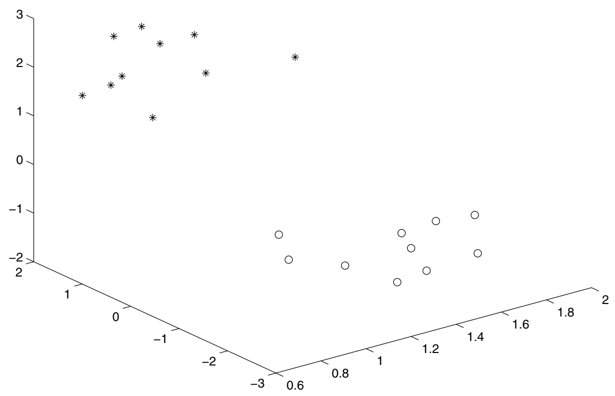
Fig. 1. Example data.

We calculated 40 trajectories of Type (2) with the data as initial points for the z_1 and z_2 sub-vectors. The z_3 and z_4 coordinates were initially set to zero for these simulation results. For the function ϕ in (1) we used $\alpha = 0.5$ and in (2) we used $\gamma = 2$. The influence of the initial values and the parameters is the subject of ongoing research. A standard Runge-Kutta routine was used to integrate (2). Some typical results can be seen in Fig. 2.

All of the z trajectories converged to one of two cluster centres, giving the following centres:

$$c_1 = \begin{bmatrix} 0.9707 \\ 2.0787 \end{bmatrix}, \qquad c_2 = \begin{bmatrix} -2.1099 \\ -1.1581 \end{bmatrix},$$

which can be seen to be close to the centres of the randomly generated data.

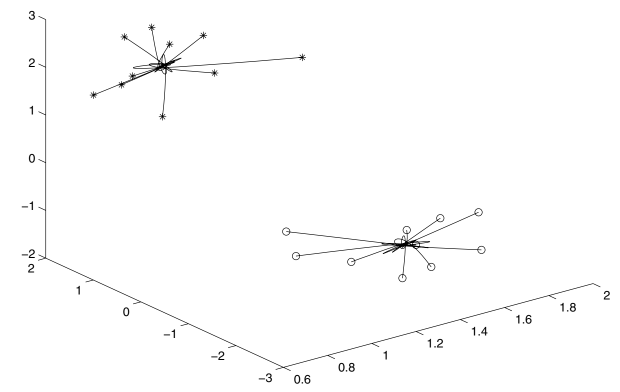
Fig. 2. Example trajectories.

We then looked for candidate points on vertical trajectories amongst the z trajectories. In order to avoid numerical problems and false candidates, we imposed some conditions on the vectors. We required the z_1 coordinate to be close to a known output value and the z_3 coordinate to be sufficiently close to zero. Also, the z_4 coordinate was initialised to zero and so we imposed $\|z_4\|$ to be sufficiently high to ensure that the probe had achieved some reasonable velocity. The actual values that we used are as follows:

- $|z_3| < 0.01$,
- $|z_1 - y_0| < 0.1$,
- $\|z_4\| > 0.3$.

Using these criteria, we found 23 candidate points for the first cluster and 28 for the second one. Using these as initial points, we then calculated the set of vertical trajectories (4). In Fig. 3 we can see the components of a typical

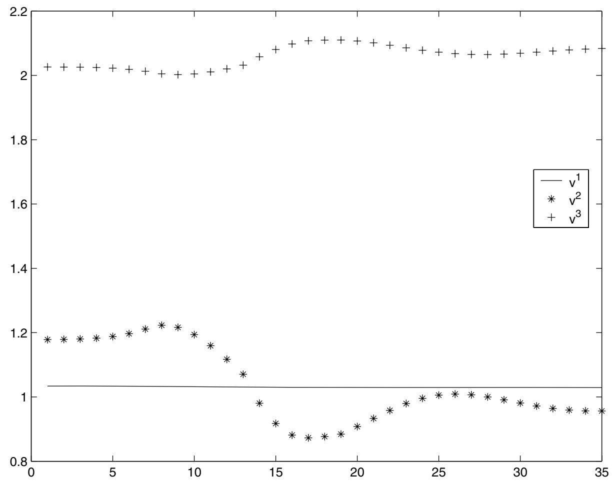

Fig. 3. Example vertical trajectory.

vertical trajectory in the time domain; as we can expect v^1 (the continuous line) remains almost constant whilst the other two coordinates vary.

The vertical trajectories were used as data to identify the parameters of the π_k functions (6). We used a standard nonlinear constrained optimisation package to carry out the calculations. We used the optimisation data set to choose the best value for α_k, and we fixed both parameters to the same value. We varied the parameter between 0.05 and 0.5 in steps of 0.05 and found that the best value for α_k, in terms of the best correlation between the real output and the predicted output for the optimisation set, was 0.05.

Then, based always on the generated vertical trajectories, we calculated $\omega^1 = 1.0426$ and $\omega^2 = 1.5069$.

In order to compare our method with another standard method for supervised learning, we applied Chiu's algorithm to the same data (Chiu, 1994). To be fair to Chiu's algorithm, we must point out that his model is similar to (8) but without the matrices A_k, and so it contains fewer parameters. However, we believe that it does give a reasonable comparison and it is a standard, universally accepted reference. We used the optimisation data to fix a value for α_k for Chiu's model and found that a value of 0.2 gave the best results. The algorithm found the following centres:

$$c_1 = \begin{bmatrix} 1.4078 \\ 2.3560 \end{bmatrix}, \qquad c_2 = \begin{bmatrix} -1.8363 \\ -0.8830 \end{bmatrix},$$

which are clearly similar to the centres that our algorithm found. We note that in the original Chiu algorithm, the centres must correspond to actual data points whereas in our algorithm this is not necessarily the case.

Both models were then tested on the validation data. The correlations between predicted and real data were found to be 0.7855 for our model and 0.7722 for Chiu's model. The real and predicted values for our model can be seen in Fig. 4 and those for Chiu's model in Fig. 5. The two are similar, although the predicted values for Chiu's model are "flatter".

We finish this section by making the remark that our model is identified wholly on generated data and not on the original data, whilst Chiu's model is the opposite i.e., identified wholly on the original data.

5. Perspectives

We have presented a method of classification based on data probes. We have made the assumption that the data are organised geometrically into homogeneous or nearly homogeneous clusters. Although this is a fairly strong assumption, we do not believe it to be totally invalid.

We have related vertical vector fields and trajectories to the notion of homogeneous data. This is an area of research into which we have been looking for a number of years and we believe that it has not yet yielded up all of its potential.

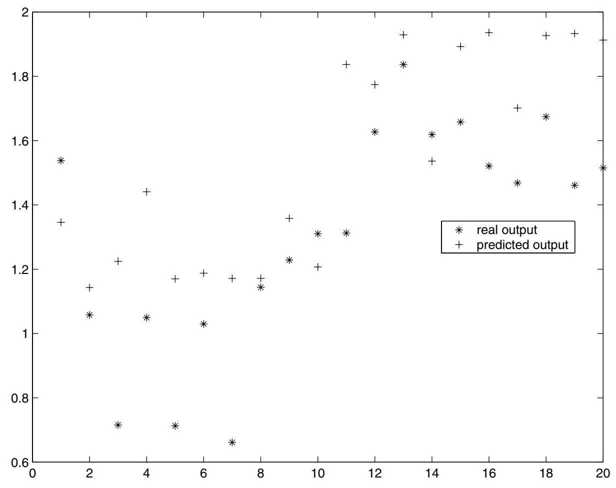

Fig. 4. Real and predicted outputs.

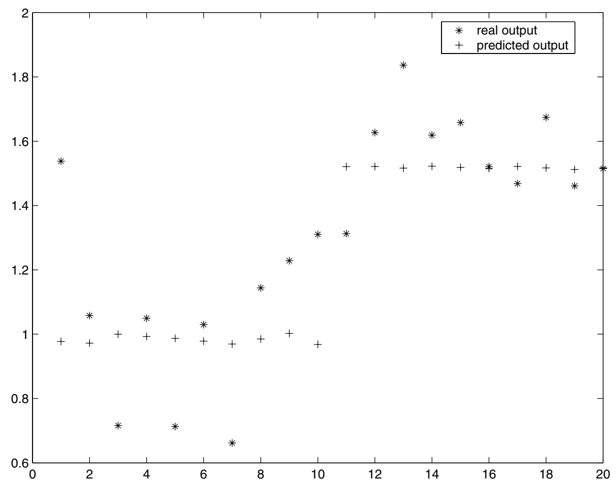

Fig. 5. Real and predicted outputs.

This work has also opened up some new areas of investigation for us. Work is ongoing to fully determine the influence of the various parameters in the model. However, it must be said that the influence of these parameters is fairly dependent on the data being analysed and so we cannot really hope to achieve any global absolute results. We are also convinced that a more sophisticated use could be made of vertical trajectories.

References

Benton T.C. and Hand D.J. (2002): *Segmentation into predictable classes*. — IMA J. Manag. Math., Vol. 13, No. 4, pp. 245–259.

Chiu S.L. (1994): *Fuzzy model identification based on cluster estimation*. — J. Intell. Fuzzy Syst., Vol. 2, No. 3, pp. 267–278.

Hand D.J., Li H.G. and Adams H.G. (2001): *Supervised classification with structured class definitions*. — Comput. Stat. Data Anal., Vol. 36, No. 2, pp. 209–225.

Hermann R. (1964): *Cartan connections and the equivalence problem for geometric structures*. — Contrib. Diff. Eqns., Vol. III, No. 2, pp. 199–248.

Hermann T. and Ritter H. (1999): *Listen to your Data: Model-Based Sonification for Data Analysis*, In: Advances in Intelligent Computing and Mulimedia Systems, (M.R. Syed, Edi.), International Institute for Advanced Studies in System Research and Cybernetics, pp. 189–194.

Isidori A. (1995): *Nonlinear Control Systems 3rd Ed.*. — London: Springer.

Pearson D.W.(1996): *Approximating vertical vector fields for feedforward neural networks*. — Appl. Math. Lett., Vol. 9, No. 2, pp. 61–64.

Pearson D.W. and Batton-Hubert M. (2005): *Increasing confidence in atmospheric pollution forecasting via vertical vector fields*. — Journal Européen des Systèmes Automatisés, Vol. 39, No. 4, pp. 553–569.

Received: 17 July 2006
Revised: 24 January 2007