ESANN'1998 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 22-23-24 April 1998, D-Facto public., ISBN 2-9600049-8-1, pp. 85-90

Parsimonious Learning Feed-Forward Control

Theo J.A. de Vries, Lars J. Idema, and Wubbe J.R. Velthuis

Control Laboratory, Faculty of Electrical Engineering, University of Twente
P.O.Box 217, NL – 7500 AE Enschede
Phone int.–31–53–489 28 17; Fax int.–31–53–489 22 23

We introduce the Learning Feed-Forward Control configuration. In this configuration, a B-spline neural network is contained, which suffers from the curse of dimensionality. We propose a method to avoid the occurrence of this problem.

1 Introduction

In order to design a conventional feedback controller, like a state-feedback controller, detailed knowledge of the process under control is required. However, in many cases an accurate process model can only be obtained with considerable effort, time and cost. Modelling difficulties are likely to arise with processes that suffer from: a) uncertainty in process structure or in many parameters, b) strong non-linearities, or c) time-variance in process structure or in many parameters. With these kinds of processes, conventional controllers and even non-linear or adaptive controllers can show poor performance because their design relies heavily on the availability of an accurate process model. Even when a reasonable accurate model is available a-priori, feedback controllers often suffer from a trade-off between high performance and robust stability.
Though feedback controllers are very capable of compensating disturbances, with good process knowledge available *feed-forward* control could be given preference over feedback control because feedback controllers are basically error driven. A feed-forward controller may be able to prevent control errors, because its output is based on the reference, instead of the error signal. When no accurate process model is available, *learning* feed-forward (LFF) controllers should be considered. LFF controllers hardly suffer from the mentioned trade-off and are capable of improving the performance of a feedback control system considerably [9]. They are not necessarily based on a physical process model and are potentially able to learn and reproduce an ‘arbitrary’ continuous function to any desired degree of accuracy, even if it concerns non-linear and/or time-variant functions.
In this paper, we propose a method to deal with the curse of dimensionality [1, 4, 5] in the context of LFF. We first introduce the LFF configuration in section 2. Next, we discuss the B-spline network that is contained in this configuration and formulate the problems that are brought about by the curse of dimensionality. In section 4 we take a look at ASMOD [4, 5, 1], which is an effective way to avoid the curse of

dimensionality when doing neural modelling. We show how the same underlying concept can be exploited in the context of LFF in section 5.

2 Learning Feed-Forward Control

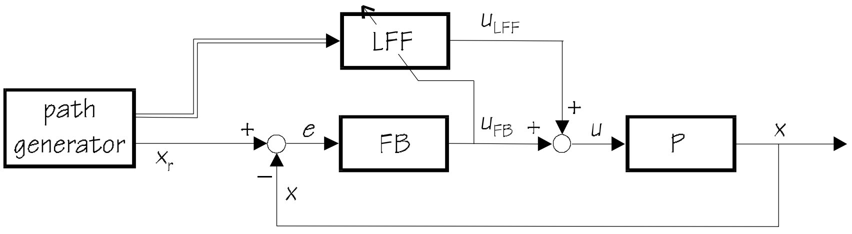

Fig. 1 Learning Feed-Forward controller configuration

We focus on learning control of motion systems, i.e., systems in which an end effector that is contained in the plant (P in Fig. 1) has to track a reference path. The learning controller considered here consists of a feedback (FB) controller and a learning feed-forward component (LFF) in parallel, as shown in Fig. 1. The feedback controller is implemented as a regular P(I)D controller and is designed for robustness. The LFF component is chosen to be implemented as a B-spline network because of its ability to combine fast learning with performance at small computational cost [2]. The learning mechanism is based on the output of the feedback controller, thus LFF control can be considered a variant of the *Feedback Error Learning* structure [6]. That is, learning is done in a closed loop, and care has to be taken in order to guarantee convergence. Unlike ‘standard’ Feedback Error Learning, the learning component in LFF is indexed by the reference path, and not by measurements of the actual path of the plant. An important reason for doing this is that the reference path is noise-free. Several successful applications of learning feed-forward control have been realised, such as:

– Path tracking for an autonomous mobile robot [9];
– Accurate positioning with a linear motion motor system [7];

3 B-spline Network

The learning component of the studied learning controller is implemented as a B-spline network [2]. B-spline networks are part of a class of networks that employ basis functions, commonly organised in a lattice structure, to approximate a continuous, possibly multidimensional, function. They can be compared to, e.g., Radial Basis Function (RBF) networks, which use Gaussian functions as their elemental basis function [8].

B-spline networks with more than one input employ multidimensional B-spline functions, formed from the tensor product of univariate basis functions, to obtain interpolation in all dimensions. With multidimensional networks, the network output is calculated as follows:

$$y(x) = \sum_{\substack{0 \le i_1 \le n_1 \\ \dots \\ 0 \le i_k \le n_k}} m_{i_1 \dots i_k}(x) \cdot w_{i_1 \dots i_k} \qquad (1)$$

where k is the number of inputs, $n_1 ... n_k$ is the number of splines defined on inputs 1...k, $m_{i_1 \dots i_k}$ is the function evaluation of the multi-dimensional B-spline function (the membership) and $w_{i_1 \dots i_k}$ are the network weights.

Compared to Gaussian functions, which are often used in fuzzy logic systems, B-spline functions have a compact support or *spline width*. Therefore, only p network weights instead of n ($p << n$) with Gaussian functions have to be updated per training epoch, p being the order of the spline functions. This property implies a considerable reduction of computational effort when employing B-spline networks in control applications. By choosing second-order B-splines on one input, linear interpolation between adjoining spline knots, i.e. the points with membership value 1, is obtained, resulting in a piece-wise linear network output function.

To adapt the weights, the back-propagation learning rule is applied:

$$w_{k+1} = w_k + h \cdot e \cdot m(x) \qquad (2)$$

with h learning rate;
e learning signal.

In the applied controller configuration (Fig. 1) either the control error e or the PID output signal can be chosen to serve as a learning signal for the B-spline network. Taking the PID output as a learning signal will result in faster learning behaviour.

Just as with RBF networks or other networks that possess a lattice architecture, the principal drawback of multidimensional B-spline networks is that the number of network weights explodes exponentially with the dimension of the input space if the basis functions are chosen uniformly. If n_i basis functions are used along each dimension, the number of network weights is given by $n_{tot} = \prod_{i=1}^{k} n_i$. Since accuracy considerations may require large values of n_i, e.g. when a highly non-linear function is to be mapped, these networks are impractical when the number of inputs k is large. With learning networks, this so-called *curse of dimensionality* brings about the following problems [1, 9]:

1. *Large number of network weights*; if the non-linear components of the process dynamics are strong, a highly non-linear function has to be mapped by the B-spline network. A network lattice that is able to map these non-linearities accurately would take up a great deal of computer memory. In practice, memory resources are limited. Therefore, network complexity and mapping accuracy have to be traded off against each other.
2. *Large training sets*; when learning a certain trajectory, only the network weights indexed by the reference path are updated. With a high number of network weights, a very large training set would be required to train the network for every possible motion through the input space. This will lead to large training times if the network is trained on-line, i.e. during control.

3. *Poor generalising (interpolating) ability*; the compensation of a non-linearity may require narrow splines for reasons of accuracy, but unfortunately, with narrow splines very different network output signals may occur with trajectories that are ‘alike’. Hence a large amount of data would be needed to fully train the network. With lattice-based networks, this trade-off between network resolution and generalising ability is inevitable.

In the next section, the consequences of these problems for the design of learning feed-forward controllers will be focused on.

4 Avoiding the curse of dimensionality through parsimony

When confronted with the curse of dimensionality, *data reduction* and *parsimonious modelling* are becoming important issues in the design of multivariate learning networks. According to the principle of parsimony, the best models are obtained using the simplest possible, acceptable structures that contain the smallest number of parameters [1]. In general, parsimony is obtained by either of the following strategies:

I. Reduce the number of input axes (network dimensions);
II. Reduce the number of basis functions, i.e. make them as wide as possible;
III. Locate centres of basis functions such that the smoothness of the required mapping is best approximated.

A technique that has shown to be able to construct parsimonious B-spline networks is Adaptive Spline Modelling of Observational Data (ASMOD) [4, 5, 1]. This technique makes use of the ANalysis Of VAriance (ANOVA) representation of a function $f(\cdot)$, with $y = f(x)$:

$$y = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{i,j}(x_i, x_j) + \ldots + f_{1,2,\ldots,n}(x) \qquad (3)$$

where $f_i(\cdot)$, $f_{i,j}(\cdot)$, *etc*. represent univariate, bivariate, *etc. additive* components of $f(\cdot)$. ASMOD is concerned with automatic iterative construction of B-spline networks according to the ANOVA representation on the basis of observational data (a training set). ASMOD is restricted to the creation of univariate and bivariate subnetworks, i.e. subnetworks with no more than one or two inputs.

The ASMOD algorithm constructs a network model by adding additional univariate or bivariate subnetworks and inserting new spline knots amongst the set of univariate knots of the existing model (self-organisation). It is also possible to delete basis functions (pruning), split bivariate subnetworks into univariate models and remove completely redundant inputs (strategies I and II). The refinement process stops if adequate approximation (in terms of MSE) is achieved. ASMOD automatically generates parsimonious models by decomposing high-dimensional input spaces into several low-dimensional subspaces. Therefore, it is an advantageous technique in case a high-dimensional data set and no information about the underlying process structure is available. The ASMOD algorithm is not concerned with *optimal placement* of the B-spline knots (i.e. strategy III).

Unfortunately, the refinement process of ASMOD starts with networks containing one spline only, and cannot be *initialised* with a prototype network. Every single combination of submodels, amount of basis functions and even order of B-spline functions is evaluated in the iteration process subsequently. This makes ASMOD a computationally intensive algorithm for highly non-linear data sets such as in our applications. Secondly, ASMOD is concerned with neural modelling from given data, whereas in our context the learning component needs to approximate an unknown control signal. These points make straightforward application of ASMOD for our purposes impossible. However, the basic concept, namely exploiting the ANOVA representation that might be contained in the network, is very useful.
With mechatronic systems, completely automatic identification an ANOVA representation is usually not necessary, as some knowledge about the process structure is available in many cases. This knowledge may be sufficient to perform an educated guess about, e.g., the desired network structure or the amount of splines on each network input domain. This idea is pursued in the next section.

5 Learning feed-forward controller based on additive networks

In practice, a-priori mechatronic plant knowledge may include one or more of the following properties:

1. *Relevance of variables*; the plant dynamics is known to be dependent on position, velocity and acceleration mainly;
2. *Dependency on input variables*; a part of dynamic behaviour can be described by univariate functions only, however, some components depend on more than one input variable.
3. *Degree of non-linearity with each input*; the shape of non-linearities may be known qualitatively for some effects; e.g. in a motion system on the velocity domain due the friction can be expected the have a Stribeck curve;

On the basis of this knowledge, initial guesses for the network inputs, the network structure, the number of B-spline functions or the B-spline distribution for each input can be made. An ASMOD-alike model decomposition is carried out ‘by hand’ in order to obtain network parsimony. However, training experiments have to be designed with care in order to guarantee that a particular effect is stored in the appropriate network.
Consider for example a linear motor motion system [7]. The dynamic behaviour of a linear motor is known to contain the following elements: force ripple, Coulomb friction, translator mass, and possibly viscous friction, stiction and reluctance. Apart from reluctance, each of these elements depends on only one variable of the reference path (x_r, v_r, a_r), and all effects result in a force that ‘disturbes’ the functional driving force. Hence, additive univariate subnetworks can be applied to compensate for the system dynamics. On the basis of this qualitative process knowledge, the controller configuration of Fig. 2 can be built. Ideally, the forces needed to compensate force ripple (a position related phenomenon) are learned by the position network, and the frictional behaviour is learned by the network that has velocity as an input. The forces needed to accelerate the translator are to be compensated by the acceleration network.

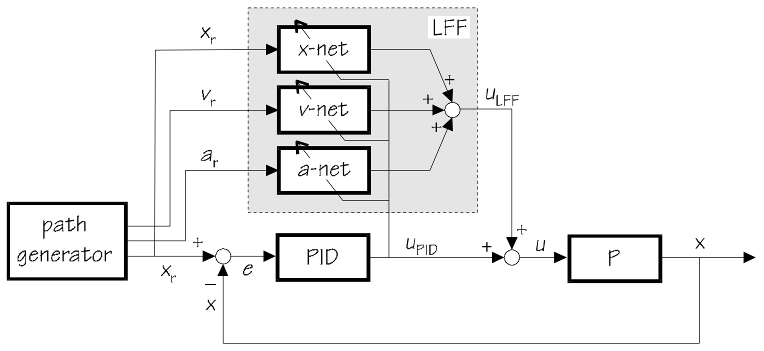

Fig. 2 Network structure used with linear motor, model parsimony and data reduction are obtained by employing additive subnetworks, which are labeled after their input.

The LFF was constructed by subsequently adding univariate networks, starting with the network designed to compensate for non-linearities expected to form the largest part of dynamics. With the linear motor system, the first network to be trained was the position network, because force ripple is the largest non-linearity. After the position network has been fully trained, a velocity (friction) network and an acceleration network were added and trained simultaneously. Experimental evaluations have shown that indeed this method works and helps to resolve the problems mentioned above [3].

6 References

1 BOSSLEY, K.M., FRENCH, M., ROGERS, E., HARRIS, C.J., and BROWN, M.: 'Recent advances in neurofuzzy control system theory and design', *Proc. ViCAM*, Euro Conference in Focused Aspects of Mechatronics (EFAM), Guimarães, Portugal, Sep. 1996, pp. 117-133,

2 BROWN, M., and HARRIS, C.: *Neurofuzzy adaptive modelling and control*, Prentice-Hall Int. Ltd., Hemel-Hempstead, UK, 1994

3 IDEMA, L.J., 'Design of parsimonious learning feed-forward controllers', MSc thesis, Control Laboratory, University of Twente, Enschede, Netherland, 1996

4 KAVLI, T., and WEYER, E.: 'On ASMOD, an algorithm for empirical modelling using spline functions', in: Hunt, K.J., Irwin, G.R., and Warwick, K. (eds.): *Neural network engineering in dynamic control systems*, Springer Verlag, Berlin, Germany, 1995, pp. 83-104

5 KAVLI, T.: 'ASMOD, an algorithm for adaptive spline modelling of observation data', *Int. Journal of Control*, 1993, **58**, (4), pp. 947-968

6 KAWATO, M., FURUKAWA, K., and SUZUKI, R.: 'A hierarchical Neural Network model for control and learning of voluntary movement', *Bio. Cybern.*, 1987, **57**, pp. 169-185

7 OTTEN, G., VRIES, T.J.A. DE, AMERONGEN, J. VAN, RANKERS, A.M., and GAAL, E.W.,: 'Linear motor motion control using a learning feedforward controller', *IEEE/ASME Tran. Mechatronics*, 1997, **2**, (3), pp. 179-187

8 POGGIO, T., and GIROSI, F.: Networks for approximation and learning, *Proc. of the IEEE*, 1990, **78**, (9), pp. 1481-1497

9 STARRENBURG, J.G., LUENEN, W.T.C. VAN, OELEN, W., and AMERONGEN, J. VAN: 'Learning feedforward controller for a mobile robot vehicle', *Control Eng. Practice*, 1996, **4**, (9), pp. 1221-1230